A HIGHLY RELIABLE, LOW POWER CONSUMPTION, LOW-COST MULTISENSORY BASED SYSTEM FOR AUTONOMOUS NAVIGATIONAL MOBILE ROBOT

Abrar Alajlan

Under the Supervision of Prof. Khaled Elleithy

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

November, 2016

A HIGHLY RELIABLE, LOW POWER CONSUMPTION, LOW-COST MULTISENSORY BASED SYSTEM FOR AUTONOMOUS NAVIGATIONAL MOBILE ROBOT

Abrar Alajlan

Under the Supervision of Prof. Khaled Elleithy

Approvals

Committee Members

Name	Signature	Date
Prof. Khaled Elleithy		11/21/16
Prof. Junling Hu		11/18/16
Prof. Xingguo Xiong		11/18/2016
Prof. Miad Faezipour		11,18,2016
Prof. Magdy Bayoumi		11/30/16

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy 12/6/16

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood 12-8-2016

Dean, School of Engineering

Dr. Tarek M. Sobh

A HIGHLY RELIABLE, LOW POWER CONSUMPTION, LOW-COST MULTISENSORY BASED SYSTEM FOR AUTONOMOUS NAVIGATIONAL MOBILE ROBOT

A HIGHLY RELIABLE, LOW POWER CONSUMPTION, LOW-COST MULTISENSORY BASED SYSTEM FOR AUTONOMOUS NAVIGATIONAL MOBILE ROBOT

ABSTRACT

There has been remarkable growth in most real-time systems in the area of autonomous mobile robots. Collision-free path planning is one of the critical requirements in designing mobile robot systems since they all featured some obstacle detection techniques. This work focuses on the collaborations of low cost multi-sensor system to produce a complementary collision-free path for mobile robots. The proposed algorithm is used with a new model to produce the shortest, and most energy-efficient path from a given initial point to a goal point. Multiple sensors are utilized together, so the benefits of one compensate for the limitations of the other. The experimental results demonstrate that the robot is capable of measuring different distances to obstacles in unknown environments. Moreover, this work aims to minimize the energy consumption of a wheeled mobile robot in dynamic environments. The total energy consumption is evaluated in multiple directions, where both motional energy and operational energy are considered, while the robot is moving in dynamic environments and avoiding collisions. A time complexity analysis

and a comparison of the proposed model, and states-of-arts methods are presented by using required resources and the overall performance of the proposed model. The proposed model is characterized by its low cost, low power consumption, and its efficiencies to follow the shortest path while avoiding collisions.

DEDICATION

To my father (Mohammed Alajlan), my mother (Lolwah Alrasheed), my husband (Sultan), and my beautiful daughters (Hala and Sara)! For all of your love and support

ACKNOWLEDGEMENTS

My thanks are wholly devoted to God who has helped me all the way to complete this work successfully.

I would like to express my sincere gratitude to my advisor Professor Khaled Elleithy for his continued support during my graduate study at the University of Bridgeport, for his guidance and boundless patience. His enthusiasm for my topic, tremendous expertise and his mission for providing high-quality work, has made a deep impression on me.

I would also like to express my special gratitude to Prof. Magdy Bayoumi, external member of the dissertation advisory committee, for his time and support. Furthermore, I am deeply grateful to the members of my committee for their positive comments, advice, and guidance: Prof. Junling Hu, Prof. Miad Faezipour and Prof. Xingguo Xiong.

I owe my deepest gratitude to my mother and father for their endless love that has always been my strength . Without them, I would not have been able to complete much of what I have done. I feel a deep sense of gratitude for my husband for his eternal support, patience, and sacrifice, this will always remain my inspiration throughout my life. I am also thankful to my beautiful daughters Hala and Sara, whom I love with all my heart for being a source of unending joy and love. You have been told “not right

now, mommy is working" often –too often, and you both have been amazingly understanding throughout the dissertation process and have been awaiting for the day I would be done. I am also very much grateful to all my family members for their constant inspiration and encouragement. Furthermore, my acknowledgements are due to my dearest friend Marwah Almasri, for being a true friend since we met at the very first day of school in 2012, for all the emotional support, caring she provided and most importantly for becoming a lifelong friend.

ACRONYMS

CA	Collision Avoidance
WSN	Wireless Sensor Network
ADC	Analog to digital converter
GPS	Global Positioning System
LLA	Low-Level Abstraction
GLA	Group-Level Abstraction
HLA	High-Level Abstraction
PRM	Probabilistic roadmap
R	Roadmap
FMG	Follow the Gap Method
APF	Artificial Potential Field
VFH	Vector Field Histogram
BA	Bug Algorithm
ROI	Region of interest
L.ref	Left infrared reflective sensor
R.ref	Left infrared reflective sensor
S	Motor Speed

Thr	Threshold
T	Time
US	Ultrasonic Sensor
g(x)	The cost of the current path segment
D_x	The traveling distance
h(x)	Heuristic estimated cost
L.IR	Left distance measuring sensor
R.IR	Right distance measuring sensor
ADC_{level}	Analog detecting level
Out_{vol}	Analog output voltage at any distance
opt_{vol}	System operating voltage
E_m	Mobility Energy
P_l	Transformation power loss by motor
m	Robot mass
a	Acceleration
g	Gravity of earth
μ	Ground friction
P_m	Motion power
DAQ	Data acquisition

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

There has been a spurt of interest in recent years in the area of autonomous mobile robots due to their high level of performance, reliability (reliable execution of monotonous tasks), cost (transportation systems based on autonomous mobile robots can be cheaper than standard track-bound systems), and accessibility (inspection of places that are inaccessible to humans, e.g. tight spaces, hazardous environments or remote places). The applications of autonomous vehicles are expected to be widely used in urban areas, industry and airport terminals, etc. It is obvious that a range of fundamental competence is needed to ensure the usefulness of mobile robots.

Mobility is almost pointless without the ability of detecting obstacles and avoiding collisions. Collision avoidance methods are mostly applied in transportation systems such as aircraft traffic control, autonomous cars, and underwater vehicles etc. Collision Avoidance (CA) is a critical requirement in designing mobile robot systems where they all demand an obstacle detection technique.

Collision avoidance algorithms in mobile robot are categorized as global or local, depending on the surrounding environments. In global collision avoidance algorithms, the surroundings are known, and the path is pre-selected, whereas, in the local collision avoidance algorithms, the environment is unknown, and some sensors are used to detect the obstacles and avoid collision[1].

In the real world, robots need to move safely in unstructured environments and achieve their given goals despite unexpected changes in their surroundings. One of the most important tasks of a mobile robot system is to acquire knowledge about its environment. This can be done through different types of sensors taking measurements and then interpreting the measured information to a meaningful data that is to be processed at the control system. Several studies have been introduced in the literature to extract the environmental details from gathered data [2].

In order to operate an efficient CA technique, many successful mobile robot systems depend on the sensing capabilities and collision detection modules attached to the robot, in order to generate a collision-free motion [3]. An obstacle's exact location can be determined through some measurements obtained from multiple sensors reading [4]. Since a single sensor is not capable of doing the task, employing multiple sensors is important. Information from different sensors are obtained and combined to find the location of the robot, detect obstacles and avoid collisions. In order to accomplish these tasks, a robot has to have an outstanding sensory system, strong mechanical structure, and sturdy controlling system. This work demonstrates the use of low cost multiple sensors in the mobile robot for collision avoidance, where the robot attains surrounding information through the sensors' readings.

1.1 Research Problem and Scope

Collision avoidance systems in autonomous mobile robots should achieve the objective of real-time collision-free path planning in order to avoid imminent collisions. A reliable and quick detection would allow time to steer away from the

possible collision. Therefore, measuring detection time and responses time should be considered when designing a collision avoidance system for mobile robot. Given that time to avoid a collision is typically in the order of seconds, using low cost resources to reduce computational time are of paramount significance.

Several mobile robotic applications require coordination between multiple sensors in order to perform complex tasks efficiently. However, programming the robot becomes a major barrier to its deployment, since some methods require the programmer to write a software for each sensor individually. As a result, the mapping between one sensor to another is complex and difficult to compose, and the code will be entrapped with the details of coordination.

1.2 Motivation behind the Research

The rapid growth in computational data analysis practices offers valuable research that can influence the amount of sensed data available in mobile robots. The information gathered from these sensors can be used to enhance the efficiency and adaptability of the system when detecting obstacles and avoiding collisions in real-time applications. The robot's behavior and environmental conditions can be figured through raw conventional sensors' data, such as camera, infrared sensors, and ultrasonic sensors. Moreover, information that significantly enhances knowledge about the surroundings can be gained from analyzing the data of the sensors.

A new programming model for mobile robot is introduced to show how it insulates from the details of the hardware, and enable the programmers to use this model in multiple hardware with one controller.

This work is a continuation of previous research, and aims to introduce a new model for optimal path planning and obstacle avoidance in an autonomous mobile robot that simply rely on the use of low cost sensors .

1.3 Potential Contributions of the Proposed Research

The main contributions of this work is to build a framework that can be used in mobile robot systems that can act robustly and autonomously for various scenarios in different environments. The proposed model should address the following requirements:

- ***Scalability:*** Scalability in wireless sensors is a pleasing property of a system. The system has to perform the same way, even when the size and the testing environment have changed. Thus, the framework can be re-scaled as its components grow and take full advantage of it.
- ***Fault Tolerance:*** Fault detection and recovery capabilities are necessary to provide the framework the ability to be used in real, and critical situations. Sensors are prone to failures due to harsh deployment environments and unattended operations. Thus a failure in one sensor should not damage the entire system.
- ***Energy-efficiency:*** Mobile robots are limited by the finite amount of energy in the batteries they carry, since a new supply of energy while working is impossible, or at least too expensive to be realistic. Thus, the overall design of sensor networks should mainly emphasize on enhancing the overall performance in terms of reducing the power consumption.

- ***Reliability:*** One of the crucial requirements for mobile robot navigation is the safety and reliability. Reliability in robotics has been followed in multiple directions: reactivity, error recovery, uncertainty handling and sensor integration. Ideally, the collision avoidance system will insure the safety of the robot without hindering the operator during normal operation.
- ***Ease of use:*** The framework is designed to be easy to use.
- ***Real-Time system:*** Support a real-time control application to provide a collision- free path for variety of robot applications and successfully interacting with environmental changes.

CHAPTER 2: LITRETURE SURVEY OF WIRELESS SENSOR PROGRAMMING APPROACHES

Wireless sensor networks are composed of tiny embedded devices, each of which has radio transceiver to send or receive packets, processor to schedule and perform tasks, and power source to provide energy for the sensor, as illustrated in Figure 2.1 [5]. Wireless sensor networks applications contain a large number of sensors used to transmit and forward data between the sensors and the sink or base station [6]. Most often, WSN is utilized for the ease of deployment and enhanced flexibility of the network. Furthermore, it supports low cost dense monitoring of hostile environments as well as disaster relief, medical care and military surveillance [7].

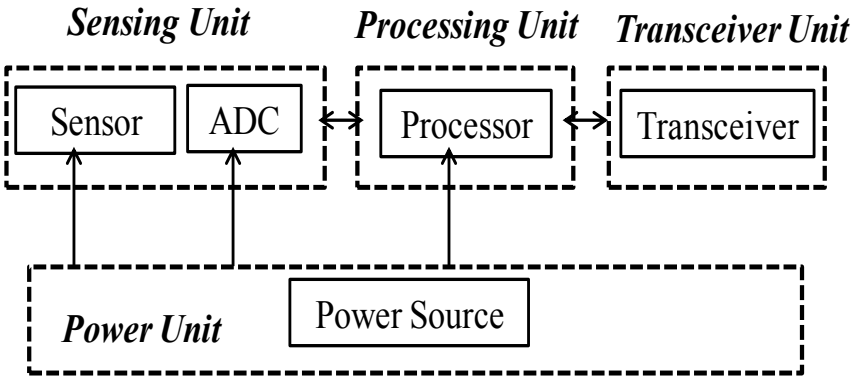

Figure 2.1: The main components of wireless sensor

The advantage of being able to place remote sensing nodes without having to run wires and the cost related to it is a huge gain. As the size of the circuitry of WSNs is

becoming smaller along with the lower cost, the chances of their field of applications are significantly growing [8]. Most sensors, depending on the requirements, are battery powered and hence conserving the energy of these sensors is very crucial. Several programming approaches have been proposed to assist WSNs programming. Two broad classes of WSNs programming models have been explored lately; local behavior and global behavior abstraction [9]. In local behavior abstractions, the application has to be programmed in details at the node-level and the programmers need to synchronize the program flow between the sensing nodes and maintain the routing code manually. In contrast, global behavior abstractions or equivalently "High-level abstraction" has emerged as one of the most important aspects in sensor networks where it is applied to hide the internal operations from system programmers. The main objective behind high-level approach is the ability to treat a group of sensors or the entire network as one single unit rather than programming each node individually [10].

The main contribution of this chapter is to provide an extensive survey on taxonomy of programming approaches for wireless sensor networks. It is also to captures some of the programming requirements that can be used for the evaluations later.

2.1 Programming requirements for wireless sensors

It is obvious that sensor networks can be used in multiple applications, that can be deployed in diverse environments. Moreover, it is very easy to modify the internal functionality of sensor networks to perform different tasks, and to support many sensor

network applications. In this section, some of the essential requirements for programming wireless sensors are expounded in detail[11].

2.1.1 Scalability

Many WSN applications deploy hundreds or even thousands of nodes collaborating to achieve desired goal(s); thus, scalability is one of the major designing attributes in sensor networks applications [12]. A scalable system refers to the ability to maintains the same when the number of sensors have changed [13]. In WSNs, scalability can be defined in two terms; size and geography. Scalability with respect to size states that if the application works properly with a few sensors, it can perform well with thousands of sensors. On the other hand, the scalability with respect to geography is defined as the ability to perform correctly in different geographical areas under different environmental conditions [13]. Since we cannot predetermine the location of sensor and we cannot assure the lifetime of wireless sensor, the programming model should help programmers in such a way to design scalable applications that are able to deliver accurate results. The location information of distributed sensors needs to be known so as to exchange the sensed data between sensors [14].

2.1.2 Localization

In wireless sensors' applications, hundreds of sensors are deployed in some areas such as underwater, or in inaccessible terrains, so their locations are random and unknown [15] , [16]. Localization in wireless sensors' applications is the determination of the geographical location of the sensor, has became one of the important aspects for wireless sensors' programming [17]. Many localization techniques have been proposed

recently, either by deploying self-localized technique or by installing a Global Positioning System (GPS) in each sensor in order to determine the exact location of the sensor. Moreover, localizing algorithms can be classified into two types:

- Range-based algorithms: each sensor is equipped with hardware measurements, so the location can be determined by calculating the distance of the selected sensor [18].
- Range-free algorithms: each sensor should determine its estimated location, and the ideal radio range of sensors.

Thus, range-based algorithms provide more information compared to range-free algorithms; however, it is more expensive since there is some hardware measuring units are attached to each sensor [19].

2.1.3 Failure-Resilience

Failure –resilience or (Fault-tolerance) is one of the most challenging requirement in programming wireless sensors applications [20]. Sensors are usually deployed in an inaccessible terrains that are unreachable by human. Some sensors might fail due to the resources limitation, hardware fault or it could be an intrusion from attackers. The failed sensors may lead to inefficient functioning of the system [21].

Thus, the system should keep performing properly even after unreliable communication, sensor failures, link failures, or unavailability of the network due to misbehaving sensor [22]. Some techniques should be adapted to alert for unexpected

failures, like monitoring the status of each sensor or using the power control technique[23].

It is a very challenging requirement for the programmers to develop a sensor application, that is robust to failures, and adaptive to the unexpected environmental changes. Moreover, providing error handling for every failure is the most challenging too.

2.1.4 Energy-Efficiency

Energy efficiency is one of the most important aspects in creating wireless sensors' applications. The overall design of wireless sensors' system should emphasize on enhancing the overall performance in terms of reducing the power consumption. The total lifetime of a battery-powered sensor is limited by the non-rechargeable batteries capacity, each sensor is equipped with a limited computation processor to perform its task [24]. Energy efficiency is very important factor in developing wireless sensors applications especially for continuous monitoring applications such as disaster monitoring, military surveillance and remote patient monitoring, etc. [25],[26].

2.1.5 Collaboration

Wireless sensors applications are vary in term of size and the number of sensors, from large scale to the small ones. Multi-sensors' application need to communicate in such a way so that the data from these sensors are gathered and analyzed at the control system. Thus, all sensors have to cooperatively and effectively work together to complete the desired tasks [27] [28].

Collaboration in wireless sensor applications can be achieved into two stages:

- Data collection type: data is collected and sent to the main server such as habitat and environmental monitoring applications.
- Collaborative information processing: converts the data gathered from multiple sensors to a higher-level information such as a tracking system applications [29].

Collaboration is not an independent requirement, it can supports other requirements. For instance, collaboration between sensors may reduces the failure-resilience where the sensing process remains functional even after one sensor failed. Moreover, collaboration inside each group may reduces the amount of data transmitted and then lower power consumption [22].

2.2 Programming Approaches for Wireless Sensors Network: A Taxonomy

In this section, a taxonomy of the programming approaches for wireless sensors is presented. Figure 2.2 depicts The entire taxonomy that categorize the wireless sensors programming approaches into low-level and high-level programming models [30]. Low-level approach mainly focuses on the use of an existing programming language to provide flexible controls over sensors.

TinyOS is a well-known example that falls into this subclass. The virtual machine that runs on each sensor is one of the interesting approaches in this subclass. It is responsible for breaking tasks and dynamically distributing them to each sensor [31][32].

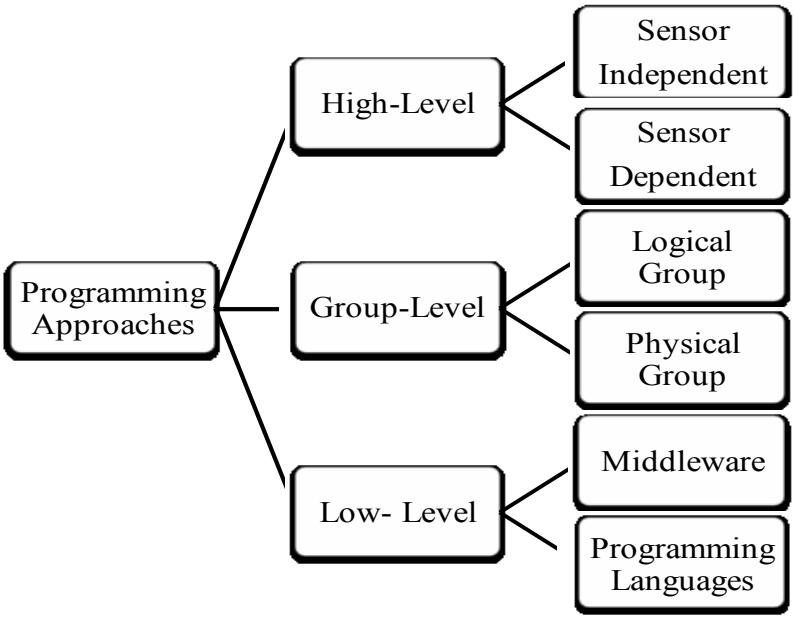

Figure 2.2: A Taxonomy of programming approaches for wireless sensors

High-level programming approach mainly focuses on simplifying the collaboration between sensors. One approach is to divide the whole system into a set of groups and treat each group as a single entity. It is called "Group-level abstractions", that helps programmer to describe collaborative algorithms easily. This approach is further divided into physical groups and logical groups. In physical group, the system can be grouped based on the physical location of the sensor, whereas the logical group is based on the shared properties among sensors.

The other approach of high-level abstraction is macro-programming abstractions, where the whole system is treated as a single entity. It is an application centric-view, thus, it helps the programmer to focus on the programming logics rather than programming the platforms. Macro-programming approach is divided into two subcategories; sensor-dependent and sensor-independent and we will cover each of them in the next sections.

2.2.1 Low-Level Abstraction (LLA)

Low-level programming approach focuses on the use of an existing programming language and abstracting hardware to provide a flexible control over the sensor.

2.2.1.1 Programming Languages

Application development at the low level is basically relying on the use of an existing programming language. NesC and C are the most well-known programming languages that are used for tiny embedded systems [26].

2.2.1.2 Middleware

The key concept behind using middleware is to support the overall performance of applications and to connect the application layer with hardware and operating systems as shown in Figure 2.3.

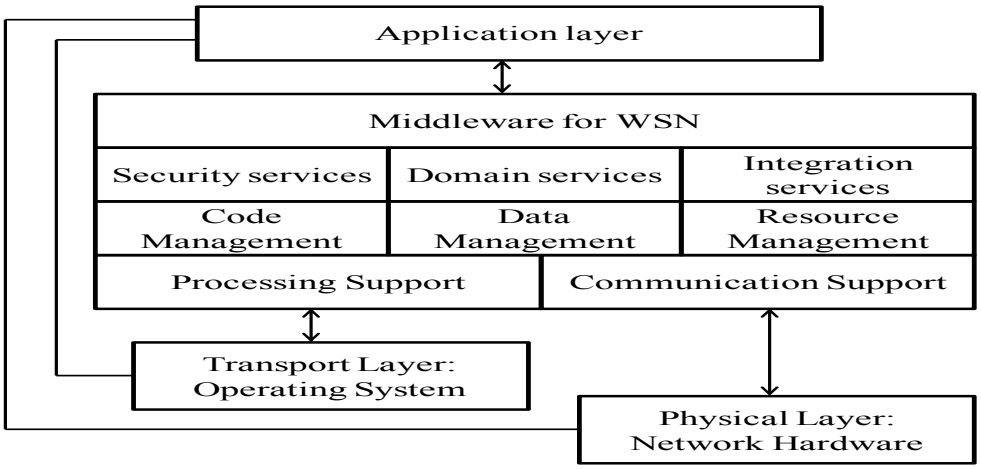

Figure 2.3: Reference of wireless sensor middleware [33]

Middleware in wireless sensors supports “re-programmability” which is the ability to break tasks and distribute these tasks to each sensor dynamically [33].

Middleware helps programmer to focuses on the programming logic without considering the implementation details at the lower level. Moreover, middleware provides a reusable code, thus, the programmer can execute a new application without using complex and inefficient methods. Furthermore, it supports system monitoring and integration [34].

Modern robots are considered as complex distributed systems composed of integrated hardware and software modules. Thus, a new software services is needed to attach all of the components together in a professional pattern to insure robustness and functionality. The middleware in mobile robots has to be self-configuring, and self-adapting [35].

Orca in [36] is a middleware for developing component-based robotics. It supports the software reuse in robotic applications and implementing a distributed component based robotic systems by permitting the developer to defines the interfaces and communication mechanisms.

Virtual Machine

Virtual machine is an example of middleware where the application is written in small segments in order to distribute them through the system using tailored algorithms. Therefore, the size of the code transmitted to each sensor is reduced and the communication amount between the microcontroller and each sensor is minimized as well [37].

Mate [38], and ASVM [39] are stack oriented virtual machines that run on TinyOS. These interpreter-based virtual machines provide an application specific

virtual machine which is employed to enhance the flexibility and offer efficient programming environments.

2.2.2 Group-Level Abstraction (GLA)

The main concept behind a group-level abstraction in wireless sensors applications is to break down the whole system into small groups and perform computations on those groups rather than dealing with each single sensor. In a group-level abstraction, the system can be grouped based on the physical locations of the sensors (Neighborhood Based) or grouped logically (shared properties) [37].

2.2.2.1 Physical Group

The notion of physical group or " neighborhood based group " is basically a sensor with its neighbor's without considering sensors' properties. This technique is used to hide the communication details between the sensors, and can be used in "localized algorithms" where the interaction between participating sensors is limited to their neighbors as in [40].

2.2.2.2 Logical group

A logical group abstraction can be defined as a set of sensors that share the same properties in sensor networks such as sensor's type, sensor input, or sensors' perform the same tasks [22]. Unlike neighborhood based, the logical group is considered to be a dynamic group since it is based on the shared properties and not limited by the

physical location of sensors [41] Logical group-based programmers cannot reuse the existing sensors without reprogramming them [42].

2.2.3 High-Level Abstraction (HLA)

Several macro-programming abstractions have been introduced recently. Macro-programming systems considered to be a high-level WSN programming model as the whole sensory- system is treated as a single entity. This approach assists programmers to emphasize on improving the semantics of the program rather than studying the characteristics of the programming environments.

There are two major classes of high-level abstractions. One is a sensor-dependent abstraction which focuses defining the global behavior of the system as a collection of sensors that can be treated simultaneously in one program. In contrast, sensor-independent approach defines the system in independent way as single unit.

2.2.3.1 Sensor- Dependent Approach

Sensor-dependent approach is intended to deliver more flexibility than sensor independent. This approach allows to define the global behavior of the computation in terms of sensors and their states [43].

2.2.3.2 Sensor-Independent Approach

Sensor-independent approach or equivalently “Database approach” is one type of high-level abstractions for sensors' programming. This approach distributes the sensors

in a space in an independent way that does not reflect any obvious abstraction for sensors [43].

2.3 Analysis and Evaluation

In this section, we focus on the most important strategies that are used in each programming model to fulfill the programming requirements discussed earlier. A summary of how each level addresses these requirements is shown in table 2.1

Programmers at low-level are able to deploy some features to enhance the scalability by using low-level interfaces. Even though, these interfaces are flexible, they tend to be complex in execution operations. To maintain collaboration and synchronization in TinyOS, two components are used: configuration to connect all components together and module to perform the synchronous method as a FIFO queue. Impala uses timer event signals to manage the collaboration and synchronization between sensors, whereas Mate installs concurrency manager and scheduler to maintain these requirements. Middleware examples, deliver efficient mechanisms for system updates to support dynamic applications and offer a great energy saving.

Moreover, scalability, collaboration and data aggregation are supported through data sharing at group-level abstraction. Caching technique is used in group-level to reduce the communications between sensors and to helps in save energy.

Caching and abstract region are employed in Hood to improve the communication failures by replacing the failed data with the old cached one. However, SPIDEY uses a redundancy mechanism to avoid flooding the whole program and to limit the

propagating of information [47]. There are some components or functions attached to each programming model to improve localization: Hood uses mirror to reflect sensor's locations [48]. Abstract region on the other hand, starts with neighbor discovery where each sensor discovers the location of its neighbors [49]. As the tracked objects move in EnviroTrack, the location of participating nodes has to be known by using some functions like *Location*: avg (position) [50].

In macro-programming approach, the system has to reduce the communication between the sensors to meets the scalability requirements. Cougar and TinyDB - most well-known examples of sensor-independent approach- are push the query selection at the edge (sensor) so the gathered data is reduced. Moreover, Cougar and TinyDB extend their SQL to express continuous sensing tasks. In addition, Regiment divides the tested area to spatial regions to facilitate the localization and communication processes.

Table 2.1 : Performance evaluation of programming model for wireless sensor

Evaluation Factors / ***Programming Models***		***Scalability***	***Localization***	***Failure-Resilience***	***Energy-Efficiency***	***Collaboration***
Low Level Abstraction						
Programming Language	***TinyOS/NesC [44][45]***	Programmers implement each feature by using low-level interfaces. Flexible but tend to be complex	Variable locations can be statically compiled into the program	Restrictions allow the nesC compiler to perform the analyses such as data-race detection to improves reliability	Restrictions allow the nesC compiler to perform the analyses such as using aggressive function in lining to reduce resource consumption	Use configuration to wire interfaces from several modules together.

Middleware	***Mate [46]***	Can express a wide range of applications	Can be extended to perform localization services	Mate is concise programs that are resilient to failure Ensures the resilient to buggy or malicious capsules	Efficient dynamic code Update : small interpreter code	Support shared variables that managed by concurrency manager
	Impala [40]	Can express a wide range of applications	Use a static location for sensors	Adaptation to device failures Autonomic behavior which increases its fault tolerance	Efficient dynamic code update Eliminating duplicate components to be transmitted over system	Not supported
Group Level Abstraction						
Physical Group	***Hood [51]***	Supported through data sharing.	Mirrors to reflect location	Caching : improves communication failures by substituting the data with old cached data	Power consumption supported through data sharing	Asymmetric group definition
	Abstract Region	Supported through data sharing.	Supported at neighbor discovery stage	Caching : improves communication failures by substituting the data with old cached data	Power consumption supported through data sharing	Supported through group definition
Logical Group	***EnviroTrack [51]***	Supported through data sharing.	The location has to be known to track object	Dynamic group management and leader election	Power consumption supported through data sharing	Supported through group definition
	SPIDEY [52]	Supported through data sharing.	Static physical location is needed to initialize the sensors	Utilize redundancy mechanism	Power consumption supported through data sharing	Supported through group definition
High Level Abstraction						
Sensor Dependent	***Kairos [43]***	No evidence for support	Each sensor responsible to determine its location	Eventual consistency	Caching	Implicit express for both distributed data flow and control flow
	Regiment [53]	Purely functional language. permit the use of fold, map	Use region for the purpose of localizing sensors	Anchor " leader" is an object persists across sensors' failures	Purely functional language. permit the use of fold, map	Region streams capable of expressing groups of sensors with geographical, and

		functions.			functions	logical relationships
Sensor Independent	***TinyDB [54]***	Queries selection at (sensor) to reduce the transmitted data	Each sensor responsible to determine its location	No evidence for support	Acquisition query processor changes battery's sampling rate to lasts for lifetime	collaboration can be defined through a query
	Cougar [55][56]	Queries selection at (sensor) to reduce the transmitted data	No evidence for support	No evidence for support	In query processing	collaboration can be defined through a query

2.4 When wireless sensor networks meet robots

Mobile robots in wireless sensor networks have gained a lot of considerations recently. They become the basis of many engineering and computer disciplines. Several sensors can be mounted on autonomous robotic systems such as unmanned robotic vehicles, and aircraft traffic control to guide them around obstacles to their goal and lead them to coordinate together to accomplish their tasks more effectively. Mobile robots can be used in sensor networks to enhance the network by providing sensors deployment, power control, and failure detections. Each sensor in sensor networks is deployed to perform a specific role, some used for sensing, routing, battery control, collecting data and so on. Robot can assist in sensor networks by acting as a sink where all data is passed to/from the entire network [57].

Moreover, wireless sensor networks can also be used to assist in eliminating problems that may appear in the field of robotics such as collision avoidance, path planning, and multi-robot coordination. Ryan Luna et.al, use wireless sensors to guide a robot through the network while detecting obstacles and avoiding collisions within

the network. It is based on the use of a static wireless network for the high-level path planning of multiple planetary rovers. In this setup, the robots receive guidelines from sensors to scheme between locations, so the robots main tasks are to detect obstacles and avoid collisions [58].

A path planning method explored in [59], uses a sensor networks to provide a road maps for the robot to traverse where each sensor generates a map based on its surrounding sensed area. Then, all the sensors maps are combined in order to generate one large map that covers the entire environment and the possible paths for the robot. Thus, the robot considers all possible paths and selects the most effective one to navigate.

Besides, the current technological studies have led to the emergence of wireless sensor actor networks that are capable of examining the environment, processing data, making decisions and taking actions based on sensors examinations. In addition, sensors are reasonable for observing the environment as they are low in cost and power, whereas actors are reasonable of making decisions since they are equipped with high processing capabilities and transition power. In some cases, a mobile robot can be considered as an actor, that is, a single network entity able to perform networking functionalities such as processing data and taking actions accordingly. It composes the mechanism by which a robot acts upon the physical environment[60].

CHAPTER 3: LITRETURE SURVEY OF COLLISION AVOIDANCE AND PATH PLANNING FOR MOBILE ROBOT

The robotic motion planning field has received a considerable amount of attention over the last decade as robots are becoming a critical part of automobile industries, underwater vehicles, and airport terminals. Collision Avoidance (CA) is one of the fundamental requirements in designing mobile robot systems, where by almost all the mobile robots applications feature some kind of obstacle detection methods.

Mobile robotic systems have significant growth in human welfare, where they represent such a complex interaction of high computational processes, outstanding mechanical design, and exceptional hardware. Majority of mobile robot applications are developed to perform some operations that require an extended level of autonomy such as security and exploration, search and rescue, inspection, etc [61]. These mobile robotic applications are normally subject to the collaboration with the dynamic environment that can be described by its challenging properties. Thus, mobile robots should have the ability to model and communicate with the surroundings, in order to achieve safe motions and reliable systems [62].

In the real world, the autonomous vehicles have to operate safely in unpredictable environments and reach their destination regardless of unanticipated changes in the tested area. One of the most significant characteristics of using autonomous vehicles is to attain some information about the surroundings through different types of sensors, taking measurements, and then used to interpret the gathered information to readable data for the control system. Several studies have been introduced in the literature to extract the environmental details from gathered data.

In collision-free path, the resulting motions mainly depends on the sensing capabilities and the actual position of the mobile robot [63].In addition, obstacles exact locations can be determined through some measurements obtained from multiple sensors reading [64]. Since a single sensor is not capable of doing the task, employing multiple sensors rises in importance. Information from different sensors are obtained and combined to find the location of the robot, detect obstacles and avoid collision. In order to perform these tasks, the robot has to have an outstanding sensory system, strong mechanical formation, and sturdy controlling system.

3.1 Taxonomy of robotics and automation

we classify the robotics and automation into programming intelligence, intelligence control, and mechanical design as illustrated in Figure 3.1.

Programming and intelligence focuses on "creating machines that perform functions that require intelligence when performed by people" [64]. It is also "The study of how to make computers do things at which, at the moment, people are better" [65].

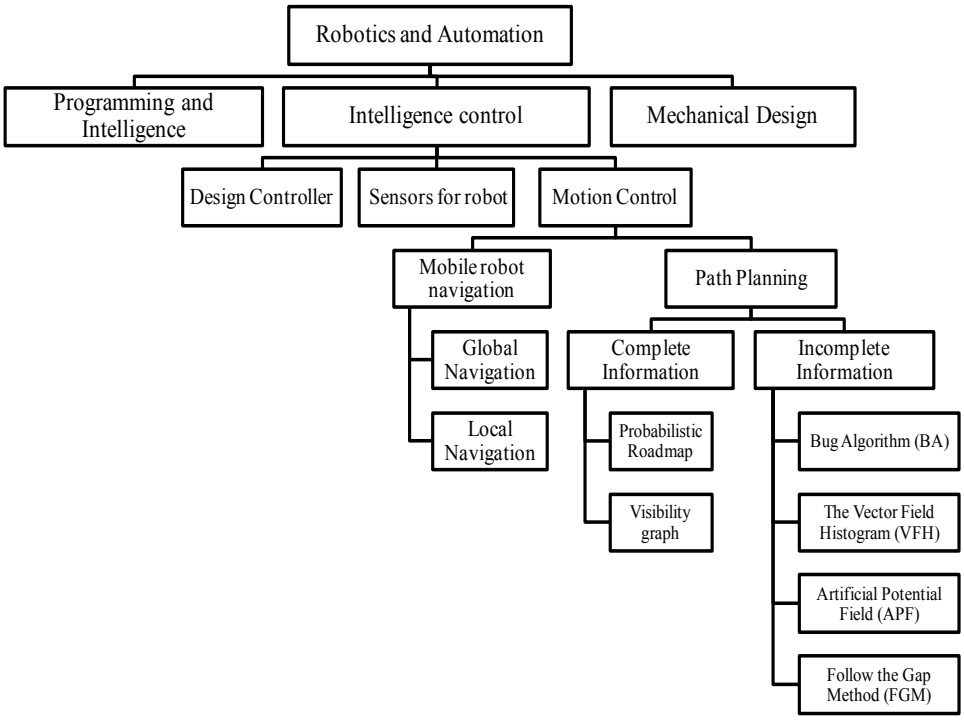

Figure 3.1: A taxonomy of robotics and automation.

Intelligent control describes the study of developing control methods that attempt to copy human intelligent behavior such as learning, planning, and making decisions based on large amounts of data [66]. Intelligent control techniques are mainly used in robotics and automation, and traffic control, where various artificial intelligence computing approaches are considered such as Neural networks, genetic algorithms, expert systems and planning systems [66]. Intelligent control is performed into three procedures: design controller, sensors for robot, and motion control.

- ***Design Controller:*** The controller serves as the brain of the system. It is responsible for planning the path and responses to environmental changes. More specifically, the controller sends signals to motor drive to proceed, based on the sensors' readings.

- ***Sensors for Robot:*** Sensors are used to measure physical quantities and convert them into readable signals for the robot. When designing a robot, it is important to choose the right sensors and build the sensory system accurately to perform the desired tasks and deliver reliable measurements.
- ***Motion Control:*** Motion control is the ability of the robot to move safely in unstructured environments and achieve given goal despite unexpected changes in the surroundings, it is further divided into two aspects: based on the navigational type and environmental type.

3.2 Mobile robot navigation

Mobile robot navigations are categorized as global navigation or local navigation,. In a global navigation, the surroundings are predefined and the path is known prior to the run. Whereas in the local navigation, the environment is unknown, and different types of modules are used to detect obstacles and avoid collisions.

3.3 Path planning

Path Planning is one of the fundamental issue in robotic systems. It is the ability to find a path for a robot from initial configuration to goal configuration without colliding with any static or dynamic obstacles in the environment. The problem of motion planning has attracted the attention of researchers due to the related complexities and the challenges of real time nature. The path planning in mobile robot can be classified into two types: path planning with complete information, and path planning with incomplete information. In the approach with complete information, all information about objects such as size, shape, and position are completely identified.

The main task is to find a path from a given initial point to goal while avoiding obstacles. Consequently, the entire operation is a one-time, off-line operation, where different optimization criteria (shortest path, low computation schema) can be performed easily. In contrast, path planning with incomplete information is formulated as an element of uncertainty is introduced and all missing information can be provided by using other sources such as sensory feedback. The beauty of this approach lies in the power of the algorithm, as it is transformed into continuous on-line operations

3.3.1 Path planning with complete information

In this approach, obstacles are static and do not change their position with respect to time. Probabilistic roadmap and visibility graph fall within this category.

3.3.1.1 Probabilistic roadmap (PRM)

The probabilistic roadmap approach to path planning consists of two phases: a constructing roadmap phase and query phase. In the constructing roadmap phase, a path from a starting configuration to the goal configuration in free space is determined. Then, connect these configurations with their neighbors, either nearby neighbors or all neighbors within a predetermined distance, so the roadmap will be constructed. In the query phase, the starting configuration and goal configuration are added to the roadmap, so that the path is obtained. The obtained path consists of three sub paths: from initial configuration to the roadmap, between the neighbor configurations in the roadmap, and from the roadmap to the goal configuration[67].

3.3.1.2 Visibility graph

A visibility graph for a set of polygonal obstacles in the Euclidean plane is a graph, whose vertices correspond to the vertices of the obstacles, and edges represents a visible connection between vertices, as the line segment connecting two vertices does not pass through any obstacles [68]. Figure 3.2 below shows the visibility graph of a set of polygonal obstacles.

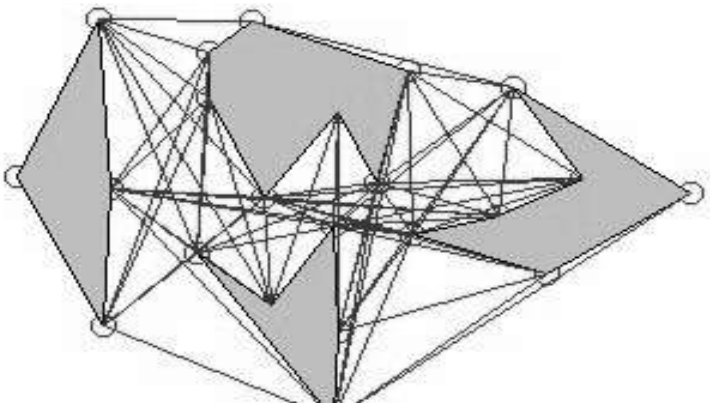

Figure 3.2: the visibility graph of a set of obstacles [68].

3.3.2 Path planning with incomplete information

In this approach, the decision is based on the current percepts captured by the sensors, as the position of obstacles may change over time. Although, several algorithms have been proposed in the literature of mobile robotics studies for performing obstacle avoidance techniques, very few methodologies have been devoted for using low cost resources and low computational power.

3.3.2.1 Follow the Gap Method (FGM)

FGM is one of the well-known collision avoidance algorithms that aims to find the gap angle between obstacles, in order to compare it with the threshold [69]. It assumes that the robot and obstacles are circular objects, and measures the robot dimension to add it into the obstacle's dimension to calculate the path . This can be done through the following processes:

- compute the gap array that stores all the distances between obstacles and the robot, in order to discover the maximum gap between obstacles.
- calculate the center angle of the maximum gap to guarantee a safe path from obstacles center.
- calculate the final heading angle by combining both gap center angle with goal angle.

After performing the entire processes, the robot can avoid obstacles and move towards the final heading angle [69].

3.3.2.2 Artificial Potential Field (APF)

APF is another collision avoidance algorithm based on the concept of potential field that has been proposed in [70] by M. Zohaib et al. In this method, obstacles generate a repulsive force to repel the robot and the target produce an attractive force to attract the robot. Figure 3.3 is taken by the work of [71] that demonstrates the obstacle's repulsive force and target's attractive force. The total force on the robot is interpreted by summing up the repulsive forces from obstacles and attractive force

from the robot. The total force is affected by how far the robot is from the obstacles and its destination.

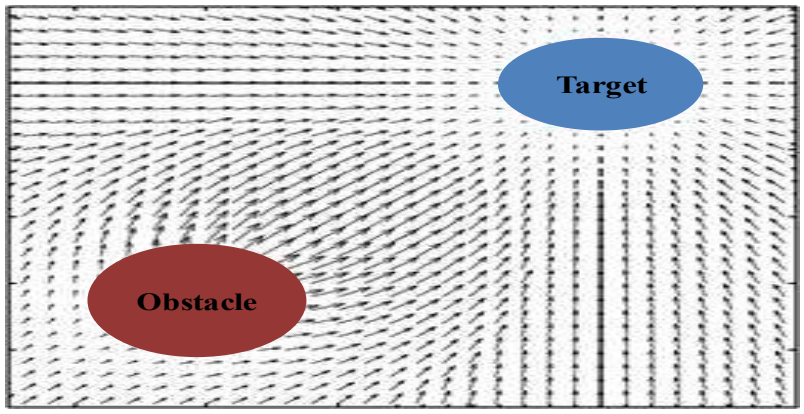

Figure 3.3: Repulsive force from obstacle and attractive force from target.

3.3.2.3 Vector Field Histogram (VFH)

VFH is another real-world obstacle avoidance method that is designed to reduce the limitations of Artificial Potential Field (APF) method [72]. This method is used to detect obstacles and avoid collision, while the robot is traveling towards its destination [73]. VFH has two-dimensional Cartesian histogram grid, and the world model that is divided into small sectors as illustrated in Figure 3.4. VFH has two stages of data reduction processes in order to compute the desired commands:

- two-dimensional histogram is converted into one dimensional histogram, and then converted to one-dimensional polar histogram.
- the algorithm selects the most appropriate sector that has a low polar density and calculates the steering angle for that direction [74].

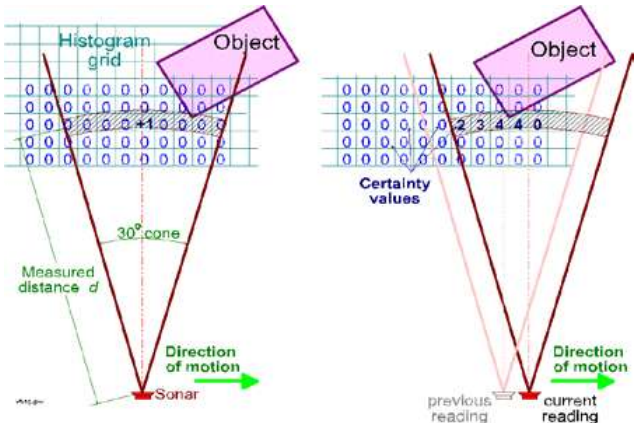

Figure 3.4: The structure of 2-dimensional histogram grid in VFH [73]

3.3.2.4 Bug algorithm (BA)

Bug algorithm, is one of the earliest collision avoidance algorithm that generate a direct path from the starting point to the final point until an obstacle is detected. During the movement in the tested environment, if the destination is not accessible, the robot stops moving to terminate the task. In bug algorithm, the decision is based on the current sensors reading without considering any previous path and previous decisions [75]. It has two behaviors:

- obstacle avoidance - the robot avoids an obstacle by tracking the edges.
- move to goal - the robot creates a reference direction to follow until the destination is achieved or an obstacle is detected.

After avoiding obstacles, the robot simply resumes traveling to its final destination without taking into consideration any other parameter [76].

3.4 Energy optimization background

One of the most challenging aspects in designing mobile robotic systems is the energy consumption, which has become a major barrier for many applications, since they are limited by a finite amount of power sources. The increase of energy consumption has created an excessive pressure on system designers to operate energy-efficient systems that are able to deliver reliable results [77]. The total energy consumption of mobile robotic applications is one of the most important issues that has not been adequately considered. Mobile robots are composed of motor drive, motors, batteries and a controller, where each of these components utilize a considerable amount of energy while operating. Moreover, the total energy consumption of mobile robot includes all energy required to keep the robot in motion, as well as the energy consumed by all modules that are used to perform a specific operation. Hence, the total energy consumption of mobile robot can be minimized by enhancing the energy efficiency of motor drives and the modules installed in the robot[78].

Minimizing energy utilization of mobile robots can be achieved in multiple directions. For example, controlling the robot's velocity, using energy-efficient modules and performing simple calculations can reduce the total energy loss on mobile robots [79]. The increase of mobile robotic applications can also support energy conservation. The study of motion planning in mobile robot can assist to preserve some energy during the robot's motion.

Even though several studies have been introduced in the literature of mobile robots motion planning, very few attention has been devoted to minimizing the energy

consumption and improving the energy efficiency [79]. Some studies are aiming to save energy by selecting the shortest path between source and destination. According to [80], minimum amount of energy is consumed when traveling distance and the number of turns are reduced. Conserving energy was also considered for uneven terrains as it heavily depends on choosing energy-efficient routes as in [81]. In their work, the terrain is modeled as grid based elevation maps where the path from source to destination is obtained using a modified version of A* search algorithm. The total energy consumption is calculated as the energy-cost of traveling from source to destination.

Mei et al in [82], produced an energy-efficient motion planning technique in open areas. They used the sixth-degree polynomial to model the cost function of energy consumption obtained in their experiments. However, this method cannot be extended to broad mission when traveling from source to destination.

Moreover, the total energy consumption of mobile robot is critically dependent on its motors speed, where energy reductions can be achieved by establishing optimal speed of the robot [83]. In [84], power conservation model for mobile robot was presented while robots were moving on a straight line with a constant velocity. Brateman et al in [85], have proposed a new technique to conserve energy in mobile robotic applications by determining motors speed and controlling microcontroller frequency. In contrast, a study proposed in [86], claims that establishing the optimal speed of the robot is not the most energy-efficient solution, as the best strategy to achieve energy efficiency is dependent on the robot's model.

In all studies aforementioned, there is no certain model that can be used to formulate the energy consumption of mobile robotic applications. The aims of this work are to provide energy-efficient dynamic motion planning model for battery-powered mobile robots. and evaluate the total energy consumption of the robot in multiple directions.

CHAPTER 4: MULTI-SENSORY SYSTEM

It is extremely important to discover the advantages and disadvantages of each type of sensors when creating a multisensory system. In some applications, operating environments can be really adverse based on the sensor's type. Since all types of sensors feature some kinds of limitations, the integration of deploying multiple sensors to take the advantages of each type is the most reliable solution. In this scheme, different types of sensors are used to create a complementary multi-sensor system that has the ability to avoid collisions and deliver reliable measurements.

4.1 Object Detection System's Description

The integration of infrared and ultrasonic sensors are used, so the benefits of one compensates for the limitations of the other.

4.1.1 Block Diagram

The mobile robot used in this work is designed to detect edges and obstacles to avoid possible collisions, based on the gathered information from multiple sensors. The block diagram of the proposed obstacle detection system is given in Figure 4.1. This mobile robot utilizes five different types of sensors: two infrared reflective sensors for edge detection, two infrared distance measuring sensors, and an ultrasonic sensor for obstacle detection. The output of the IR distance measuring sensors are analog in

nature, as they are based on the amount of reflected lights. Therefore, the analog output needs to be processed using Analog to Digital Converter (ADC) input line, in order to generate digital signals containing information about the obstacle's distance. Then, the microcontroller works as the robot's brain, it processes all data acquired from the mounted sensors, and create control signals to the motor drive to perform the required moves. Modeling of the robot's environment and the typical characteristics of various sensors used are described in details in the following sections.

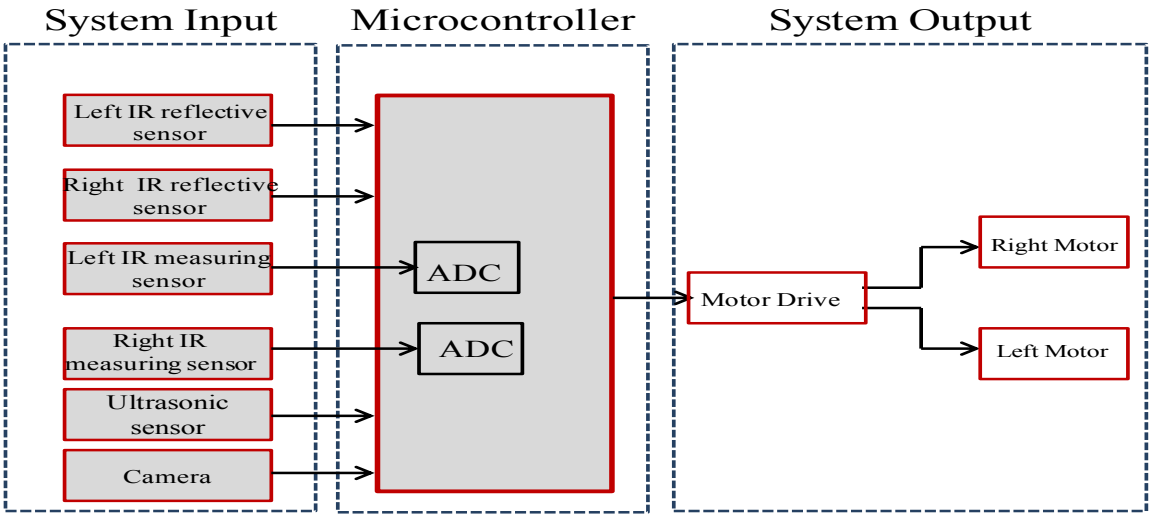

Figure 4.1**:** Block diagram of the proposed collision- free motion control

4.1.1.1 Edge Detection: Using infrared reflective Sensor

Edge detection is one of the primary behaviors in designing autonomous mobile robot. The robot change its direction when an edge is detected to avoid moving over the edge. In general, the edge is the area that does not reflect, for example; the edge of a table can be detected when the reflection from the IR beam has stopped. Therefore, in

mobile robots, edges are detected by using infrared reflectance sensors. In order to perform the task, two-TCRT5000 sensors are used as illustrated in Figure 4.2 .

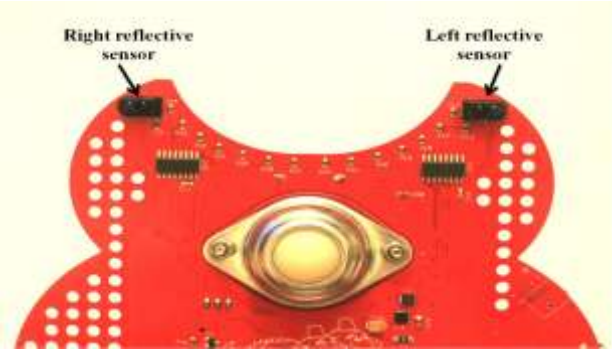

Figure 4.2: Reflective sensors on mobile robot (FEZ Cerbot)

The TCRT5000 sensor is the most suitable type for detecting lines and edges. Moreover, in some cases , it can be used for a very small range obstacle detection. The TCRT5000 measures the reflected light from the transmitted IR beam. It consists of two components:

- emitter to generates infrared light that bounces off surface .
- detector to converts the reflected light into a measurable form (digital data) as depicted in Figure 4.3 .

TCRT5000 works best when the measuring surface is a few millimeter away from the robot as the detection range is between 0.2 to 15 mm.

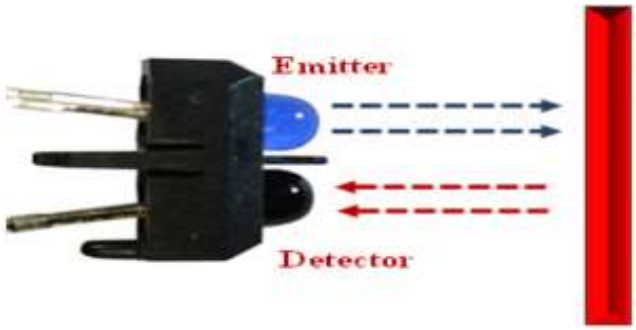

Figure 4.3: TCRT5000 infrared sensor for edge detection

Three different situations are simulated in Figure 4.4. according to the edges' positions

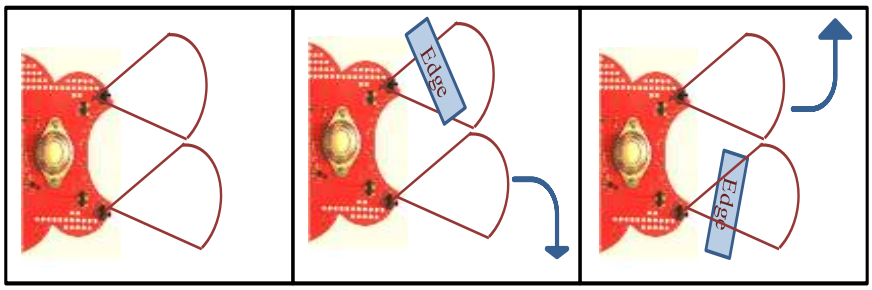

Figure 4.4: Edge detection scenarios

In the case of no edges detected, lots of reflected lights hit the sensor and the difference between left and right sensors' reading is close to zero. Therefore, both motors will move at the same speed (the robot move forward).

If an edge is detected through the left sensor, less light is reflected back to the left sensor, while the right sensor detects maximum reflection. The difference between left and right sensors' readings is positive, hence, the left motor speeds up while the right motor slows down, so the robot turns to the right.

If the edge is detected through the right sensor, less light is reflected back to the right sensor while the left sensor detects maximum reflection. The difference between left and right sensors' readings is negative, hence, the left motor slows down while the right motor speeds up, so the robot turns left.

4.1.1.2 Obstacle Detection: Using infrared Sensor

Obstacle detection is a crucial requirement for any autonomous robot, as it is designed to investigate the surrounding and detect obstacles through some distance measuring sensors. Infrared sensors are the most fitting type for obstacle detection due

to their low cost and ranging capabilities. In this work, two Sharp GP2D120 distance measuring sensors are attached to the robot to perform obstacle detection as depicted in Figure 4.5.

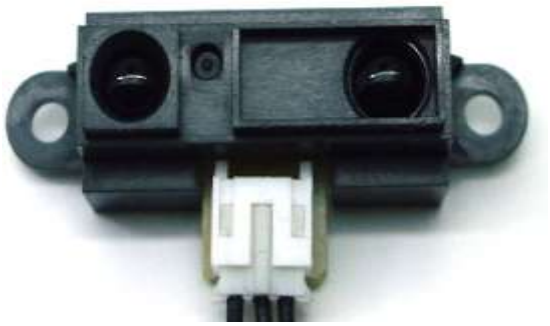

Figure 4.5: Sharp GP2D120 distance measuring sensor

Sharp GP2D120 is a small and inexpensive device that is considered as the primer infrared proximity sensor, as it uses the triangulation method to calculate the distance to object. It has a detection range between 4 to 30 centimeters with a reasonable precision and nearly impenetrable to variations of obstacles reflectivity. While the robot is moving, the IR light generated by the emitter is traveling out in the tested area and either hits an obstacle or just keeps on going. In the case of existing obstacle, the light reflected off an obstacle returns to the detector, and forms a triangle between the emitter and detector. The measurement of the distance using triangulation method is illustrated in Figure 4.6. In the case of no obstacle encountered, the light is never reflected back, and the sensor reading indicates no obstacles around.

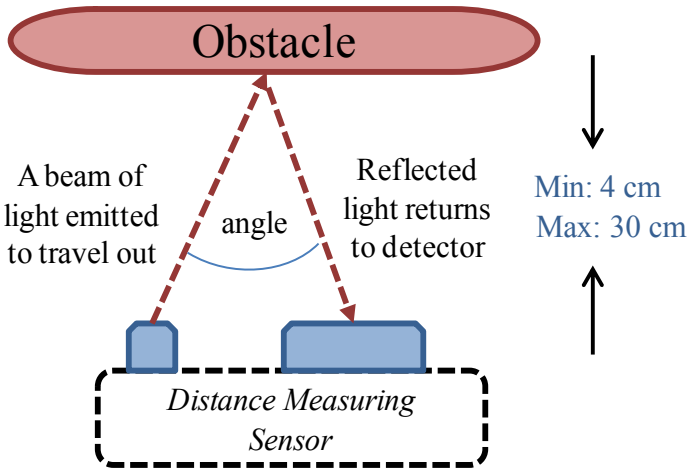

Figure 4.6: Sharp GP2D120 triangulation method

4.1.1.3 Obstacle Detection: Using ultrasonic Sensor

Ultrasonic sensor is also one of the most widely used in autonomous mobile robot to measure distances as it presents a consistent source of obstacle detection. In the case of transparent obstacles and poor lightening, the ultrasonic sensor is the most appropriate type, since it does not depend on vision. The ultrasonic sensor's basic rule is to transmit wave package and measure the time it takes to get the echo back. In the case of an existing obstacle, the wave collides with the obstacle and is bounced back to the sensor. Then, the time difference between sending and receiving waves is calculated to determine the distance to the robot.

In this work, Distance US3 module is an example of ultrasonic sensor that has been used as presented in Figure 4.7. Distance US3 module has a detection range between 2 to 40 centimeter with a measuring angle of 15 degrees.

Figure 4.7: Distance US3 Module

4.1.1.4 Obstacle Detection: Using camera

Video cameras are one of the most commonly used sensors in a wide range of applications such as object detection, tracking and recognitions. A single onboard camera has been mounted at the front of the robot to detect and track objects. The object detection process can be divided into four main procedures as illustrated in Figure 4.8, and the following four algorithmic steps:

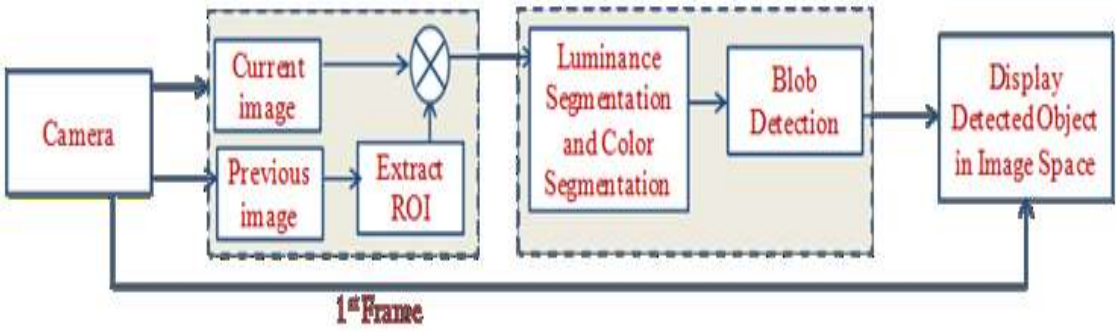

Figure.4.8 A block diagram of object detection by camera

- eliminate the effect of ego-motion of the camera and extract a region of interest (ROI). Two images of the same scene are linked by a nonsingular linear

transformation (previous image and current image), so the effect of ego motion is eliminated.

- detect foreground object and perform video segmentation. The first frame of the video is used as the background where the intensity and color information for the background subtractions are considered to improve the accuracy of foreground segmentation.
- calculate the foreground objects and connect foreground points as a blob using blob analysis block. The blob analysis block is used to calculate statistics for labeled regions in a binary image, where blob library for OpenCV is used to detect connected fore-grounded regions.
- display detected object in image space.

4.2. Obstacles detection algorithms

This section describes the proposed approach for collision avoidance method based on the integrations of several modules. The robot is able to navigate in the environments according to the outputs of the attached modules. The integration of two infrared reflective sensors, distance proximity sensors, and ultrasonic sensors are used to detect obstacles and avoid possible collisions in mobile robot.

4.2.1 Edge detection algorithm

The robot starts sensing the environments for edges through the reading of two infrared reflective sensors mounted in front of the wheels. Then, the readings will be compared with the given threshold value. If both readings are less than the threshold,

then an edge is detected. Otherwise, no edge is detected and then will move to the next stage (obstacle's detection). The detailed algorithm is shown in Figure 4.9.

Algorithm1: Edge-detection Algorithm

```
Input: L.ref: Left reflective sensor; R.ref: Right reflective sensor;  S: Motor speed; thr: Threshold; t:time;
Output: α: Edge is detected ;
BEGIN
    INIT  s   // Initialize the speed
    WHILE (true)
      READ L.ref   // Read left reflective sensor
      READ R.ref    // Read left reflective sensor
      IF ( L.ref < thr && R.ref < thr )  // Compare left sensor reading and right sensor reading with the threshold = 10)
        THEN SET α == true  // Edge is detected  if both readings are less than the threshold.
      ENDIF
    ENDWHILE
```

Figure.4.9 Edge-detection algorithm

4.2.2 Proximity Sensors-Based Distance Measurement Algorithm

In the case where no edge is detected, two of Sharp GP2D120 sensors are used and AnalogTodistance( ) is called to convert the sensors analog values to distance . If the sensors readings are within the detection range, then an object is detected. Otherwise, no object is detected, and the robot will continue following the path as illustrated in figure 4.10

Algorithm2: Proximity Sensors-Based Distance Algorithm.

```
Input: L.IR: Left measuring distance sensor; R.IR: Right measuring distance sensor; S: Motor speed; thr: Threshold; t:time;
Output: α: Obstacle is detected ;
BEGIN
    INIT  s   // Initialize the speed
     WHILE (true)
            READ L.IR // read left infrared sensor
              CALL AnalogToDistance (L.IR) // convert analog reading to digital reading
            READ R.IR  // read right infrared sensor
              CALL AnalogToDistance (R.IR)    // convert analog reading to digital reading
        IF ( (L.IR >= 4 &&  L.IR <= 30)  || (R.IR >= 4 && R.IR <= 30) ) // Compare the reading with the detection range of
            THEN  SET  α == true     // Obstacle is detected
          ENDIF
```

Figure.4.10 : Proximity Sensors-Based Distance Algorithm.

4.3.3 Ultrasonic-Sensor- Based Distance Measurement Algorithm

In the case there where no edge is detected, Distance US3 Module will be used to measure the distance from an object within 2cm to 40 cm range. GetDistanceInCentimeters() function is used to convert the senor readings into centimeters. The detailed algorithm is shown in Figure 4.11

Algorithm3: Ultrasonic Sensor -Based Distance Algorithm.

```
Input: US: Ultrasonic sensor; S: Motor speed; thr: Threshold; t:time;
Output: α: Obstacle is detected ;
BEGIN
    INIT  s   // Initialize the speed
       WHILE (true)
            READ US // read ultrasonic sensor
              CALL GetDistanceInCentimeters (US)  // convert the reading to centimeter
            IF (US >= 5 && US <= 30) // Compare the reading with the detection range of each sensor
               THEN  SET  α == true     // Obstacle is detected
          ENDIF
    ENDWHILE
```

Figure.4.11 : Ultrasonic Sensor-Based Distance Algorithm.

4.3.4 Path planning collision-free algorithm

The algorithm for the entire system and the flowchart are presented in figure 4.12, and figure 4.13 respectively. The proposed model deploys a modified version of A* searching algorithm to find the shortest path from a given point to target. It has been widely used in path finding for its reduced search space. It uses a best first search where it takes an input, evaluates a number of possible paths, and returns the least cost path from a given source to destination [87]. As the robot follows the least cost path, it keeps a sorted queue of alternate path along the way to abandon any higher cost path segment at any point of time and use the lower-cost segment instead. This procedure continues

until the target is reached [86]. The cost of the path can be calculated based on the distance traveled from a given point to another as follows:

$$g(x) = g(x-1) + Dx \quad (1)$$

where g(x-1) is the cost of the previous path segment, and D*x* is the traveling distance between *x-1* and *x*. Then, the least cost path from a start point to target can be determined based on the cost of each path segment as follows:

$$f(x) = g(x) + h(x) \quad (2)$$

where h(x) is a heuristic estimated cost of the path from current point to the goal point. The algorithm used in this work is characterized by its high searching efficiency, and completeness.

When the shortest path from initial position to target has been determined, the proposed collision avoidance model performs two processes; edge detection and obstacle detection.

At the initial position, the robot starts sensing the environments for edges through left and right reflective sensors to avoid moving over the edge. In case of possible collision, the reflective values are compared with a predefined threshold and the robot will make an action accordingly. On the other hand, if there is no detected edge, the difference between left and right sensor readings is close to zero. Therefore, both motors move at the same speed (moving forward). Then, the robot performs obstacle detection in order to avoid any imminent collisions. Two infrared distance measuring sensors; an ultrasonic sensor, and a camera are used for obstacle detection. The

microcontroller converts all sensors readings to measurable form (distances in centimeters) to compare them with their detection range. If an object is detected through any sensor, the robot spins around the object and moves forward. Thus, the microcontroller decision is based on sensors readings.

Algorithm4: Path Planning Collision-Free Algorithm

Input: L.ref: Left reflective sensor; R.ref: Right reflective sensor; US: Ultrasonic sensor; L.IR: Left measuring distance sensor; R.IR: Right measuring distance sensor; S: Motor speed; thr: Threshold; t:time;
Output: α: Edge is detected ; β: Obstacle is detected

```
BEGIN
    INIT s   // Initialize the speed
    GET source.position == start // get start point
    GET goal.position == goal   // get goal point
    WHILE (true)
      IF start == goal    // compare if the start point equal to goal point
        THEN Goal reached
    ELSE WHILE (start != goal ) DO
      ADJUST heading to goal
      COMPUTE shortest path to goal
      READ L.ref   // Read left reflective sensor
      READ R.ref    // Read left reflective sensor
      IF ( L.ref < thr && R.ref < thr )  //Compare lsensor reading and right sensor reading with threshold
        THEN SET β == true  // Edge is detected  if both readings are less than the threshold.
      ELSE
          READ US // read ultrasonic sensor
          CALL GetDistanceInCentimeters  (US) // convert the reading to centimeter
          READ L.IR // read left infrared sensor
          CALL AnalogToDistance (L.IR) // convert analog reading to digital reading
          READ R.IR  // read right infrared sensor
          CALL AnalogToDistance (R.IR)    // convert analog reading to digital reading
       IF ( (US >= 5 && US <= 30) || (L.IR >= 4 &&  L.IR <= 30)  || (R.IR >= 4 && R.IR <= 30) )
            THEN   CALL SetMotorSpeed (0,0)  //  the left motor speed and right motor speed to stop
                       SET t = 1000ms  //  set timer  for  1 sec
                       CALL SetMotorSpeed (-s,s)  //  turn left
                       READ L.IR // read left infrared sensor
                       CALL AnalogToDistance (L.IR) // convert analog reading to digital reading
                       CALL SetMotorSpeed (s,-s)  //  turn right
                       READ R.IR // read right infrared sensor
                       CALL AnalogToDistance (R.IR) // convert analog reading to digital reading
                  IF ( (L.IR >= R.IR) // if the left distance is smaller than right distance
                       THEN CALL SetMotorSpeed (s, s) // move forward
                     ELSE
                       CALL SetMotorSpeed (-s,s)  //  turn left
                       CALL SetMotorSpeed (s, s) // move forward

          ELSE
            CALL SetMotorSpeed (s,s) //  move forward  .. no object  detected
        ENDIF
      ENDIF
    ENDWHILE
    ENDIF
ENDWHILE
```

Figure 4.12: Path planning collision-free algorithm

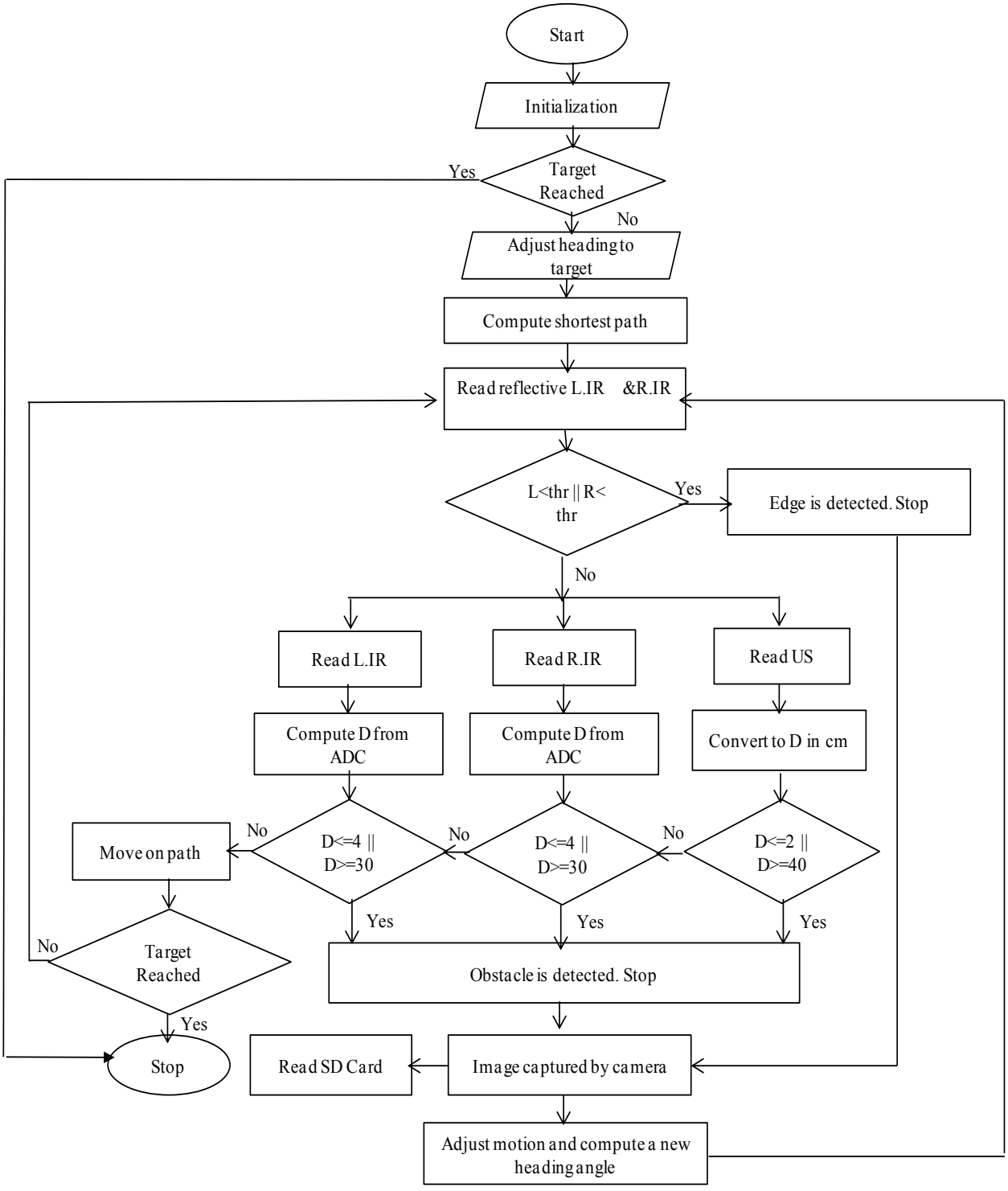

Figure 4.13: A Flow chart of the proposed motion planning approach

CHAPTER 5: MODELING OF FEZ CERBOT ENVIRONMENT

The mobile robot used in our experiments is a specially designed robotic platform for performing object detection and collision avoidance control as shown in Figure 5.1

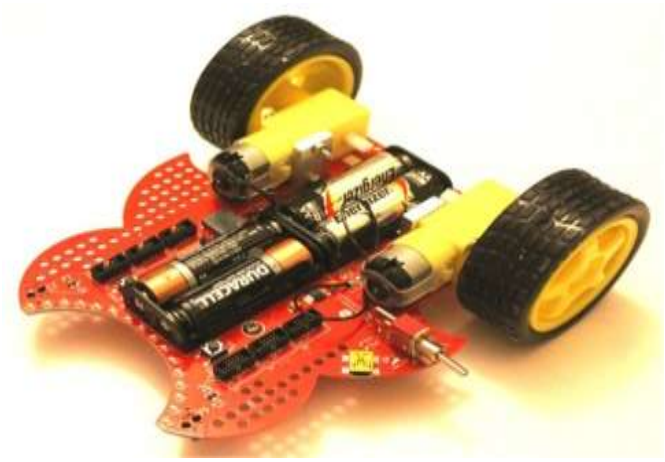

Figure 5.1. FEZ Cerbot

5.1 Design Structure of FEZ- Cerbot

It was tested in [88], by authors to perform simple collision avoidance technique without considering data analysis and sensors measurements.

For the implementation of the above motives we use .NET Micro Framework which is an open source platform for embedded devices. Using Visual Studio and C#,

developers can create embedded applications [89]. In addition, Microsoft has recently introduced the .NET Gadgeteer, which is an open-source platform that enables using .NET Micro Framework and Visual Studio for combing the benefits of object-oriented programming and the assembly of small electronic devices [90]. .NET Gadgeteer is a standardized way to connect mainboards and modules. One of the most well-known companies that adapt the use of .NET Gadgeteer and .NET Micro Framework is GHI Electronics. GHI Electronics offers a variety of mainboards, sensor modules, and power modules [91].

FEZ Cerbot is a wheeled robot that has the following specifications:

- 168Mhz CPU.
- 2 gears/motors
- 16 configurable LEDs
- 2 reflective sensors.
- four AA battery holder.
- six Gadgeteer sockets that can be used for modules.
- one USB Client cable for connecting to PC [91].

In this work , FEZ Cerbot robot from GHI is used to test our proposed method. Figure 5.2 shows the entire system where all sensors are attached including the infrared reflective sensors, infrared distance measuring sensors, ultrasonic sensor, and robotic platform.

The two TCRT5000 infrared reflective sensors are installed in the front of the robot, where they should be positioned in front of the wheels to detect an edge before a

wheel would. Moreover, these sensors have to be spaced widely as possible to assist the robot to find the best steering angle to turn when an edge is detected. According to the reflectance values, the robot microcontroller, then takes action. Since these sensors are used for edge detection, they work best when measuring surfaces that are a few millimeters away.

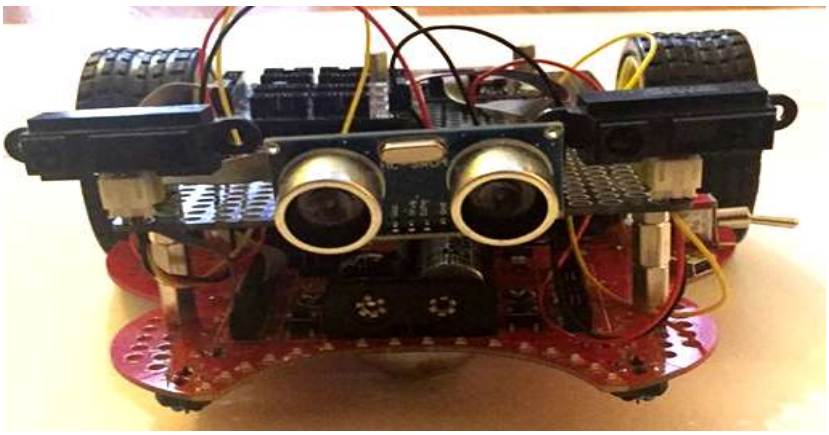

Figure 5.2: Test-bed prototype of FEZ-Cerbot

Two GP2D120 infrared sensors are mounted in the front of the robot for obstacle detection and distance measurements. This measuring sensor is composed of two parts: emitter to transmit the infrared light and a sensing device to produce the output voltage, based on the results of the triangulation method. The maximum voltage output from a GP2D120 sensor is about 3V. The maximum detection range of the GP2D120 sensor is between 4 to 30 centimeters. The GP2D120 generates an analog voltage that can be measured using analog-to-digital converter (ADC) input line.

Distance US3 module works best for distances between 2 and 40 centimeters, with a measuring angle of 15 degrees. Ultrasonic sensor is sufficient for robot

navigations, since it calculates the distance to the object by figuring out the time interval between transmitting signals and receiving echo back.

USB Camera module can stream images as large as 320x240 with up to 20fps on smaller images. Since the frames of a sequence video contain some specific information about the detected objects, a camera module is used for object detections. The ego-motion is compensated, as a big difference may occur by the motion of the camera.

The integration of the information supplied by multiple sensors can be the best solution to overcome the spatial uncertainty of unknown environments in several advanced robotic navigation applications.

5.2 Real time experiments

A set of experiments were performed on a FEZ Cerbot mobile robot in a lab environment. The robot is equipped with multiple sensors to alert if an object is detected while moving along its path. For each experiment, the robot starts sensing the environment for objects, when an object is detected, the robot adjust the motion and compute the new heading position. When an object is detected through the right infrared sensor, maximum light is reflected back to the right infrared sensor, while less light is reflected to the left infrared sensor. The difference between the right and left readings is positive, hence, the right motor speeds up while the left motor slows down, and then the robot turns to the left. On the other hand, when the obstacle is detected through the left infrared sensor, the difference between the right and left readings is negative. Thus, the right motor slows down while the left motor speeds up, and then

the robot turns to the right. The experiment was executed in multiple scenarios with different shapes of obstacles placed in multiple locations around the robot. The experiments were carried out as follow:

- Experiment 1 : The first experiment was conducted to examine the ability of the infrared sensors to detect obstacles and avoid collisions. It took a place in white surface with three large opaque red cylinders to serve as obstacles as presented in Figure 5.3.

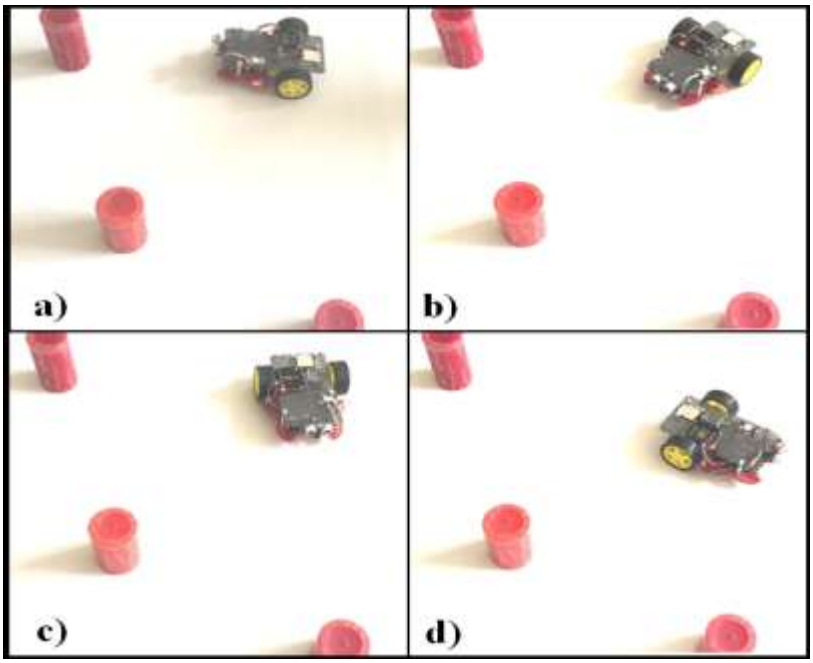

Figure 5.3: A snapshot of a real-time experiment when detecting opaque objects

- Experiment 2 : The second experiment was conducted in a parquet floor with smooth walls, where translucent and transparent objects are placed around the robot. The experiment was performed to examine the ability of the infrared distance measuring sensors and ultrasonic sensor to detect translucent and transparent objects as illustrated in Figure 5.4

Figure 5.4: A snapshot of a real-time experiment when detecting translucent and transparent objects

- Experiment 3 : The third experiment was tested in a dark area, so the ambient light does not affect the results. The experiment was run to examine how the ultrasonic sensor behaves for different light frequencies.
- Experiment 4 : The fourth experiment as shown in Figure 5.5, was conducted to examine the ability of detecting moving objects. The experiment is performed in an open area with two robots, where each robot is equipped with proximity sensor to detect each other and measure their relative distance.

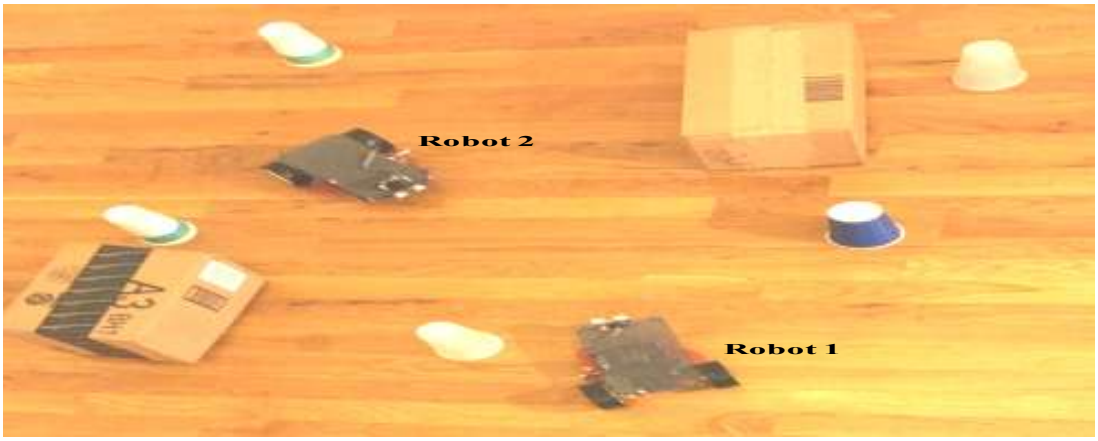

Figure 5.5: A snapshot of a real-time experiment when detecting moving objects

- Experiment 5 : The fifth experiment was performed to determine the stability of detecting objects with a moving camera. Object detection flow by camera is depicted in Figure.5.6. The performance of object detection by camera was evaluated using two videos with each having a different number of frames and objects. The final evaluation results are presented in Table 5.1, where true positive is the number of accurate detections, false positive is the number of inaccurate detections, and detection rate is represent the percentage of correct detections.

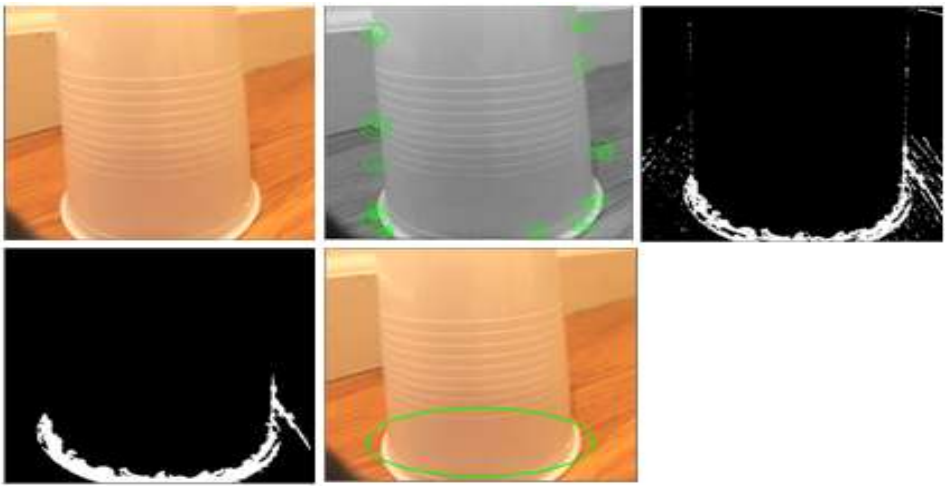

Figure.5.6.a. A selected frame from reference video. Figure. 5.6.b Feature points is detected. Figure.5.6.c. Resulting binary masks from segmented image (where abandoned objects appear) Figure.5.6.d. The resulting binary image is cleaned with morphological filtering to clean up the mask image and removing points that do not represent object. Figure.5.6.e. The result of detected object.

Table 5.1: The final evaluation results of object detection by camera

Video	*Number of Frames*	*Number of objects*	*Number of detected objects*	*True Positive*	*False Positive*	*Detection Rate*	*Completion Time*
1	27	19	17	16	1	84.21%	0.703 sec
2	76	58	51	46	5	79.31%	2.109 *sec*

- Experiment 6: Unlike to other reported algorithms, the proposed model guarantees a collision free path from simple to critical environments including U and H shaped obstacles. In the case of U shaped obstacle, the robot senses the side walls, calculates the distance to front wall, and steers inside. As the front wall is detected, the robot turns about 90° to avoid it, and then it turns about 90° again when the side wall is detected. After avoiding a U shaped obstacle, the robot continues its motion in that direction. The overall performance of the proposed model when detecting U shaped obstacle can be seen in figure 5.7

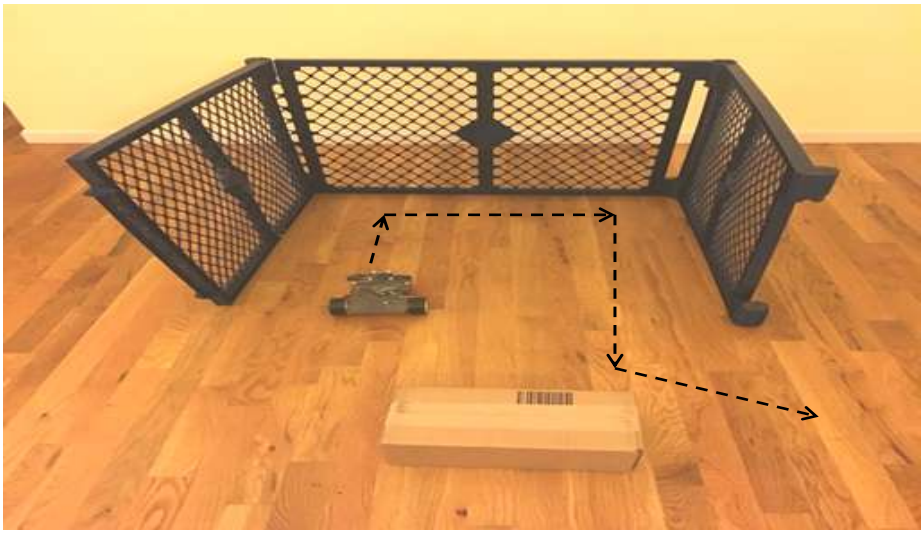

Figure.5.7. A snapshot of a real-time experiment with a robot and U shaped obstacle

- Experiment 7 and 8 : The experiments were carried out to find optimal path planning in simple and complex dynamic environments, where the robot started from a given position and required to travel toward a predefined goal position following the shortest path. The two experiments are illustrated in figure 5.8, and figure 5.9 respectively.

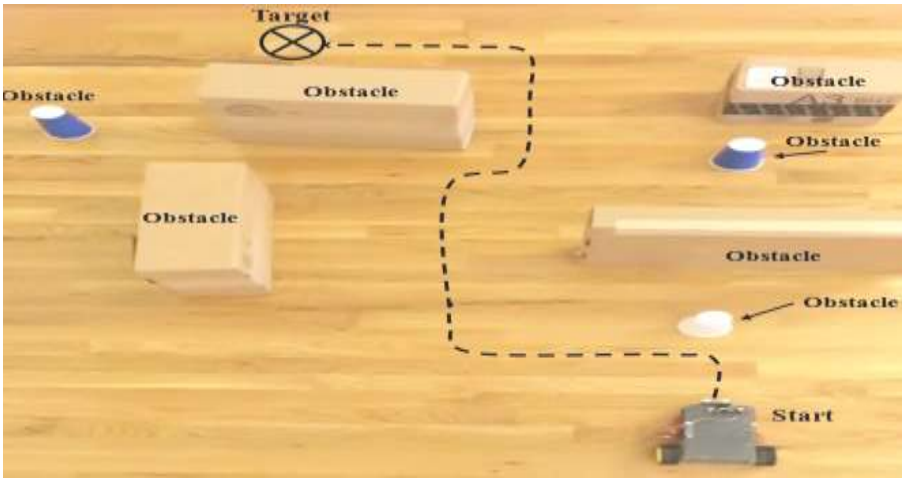

Figure.5.8. A snapshot of a real-time experiment for optimal path planning

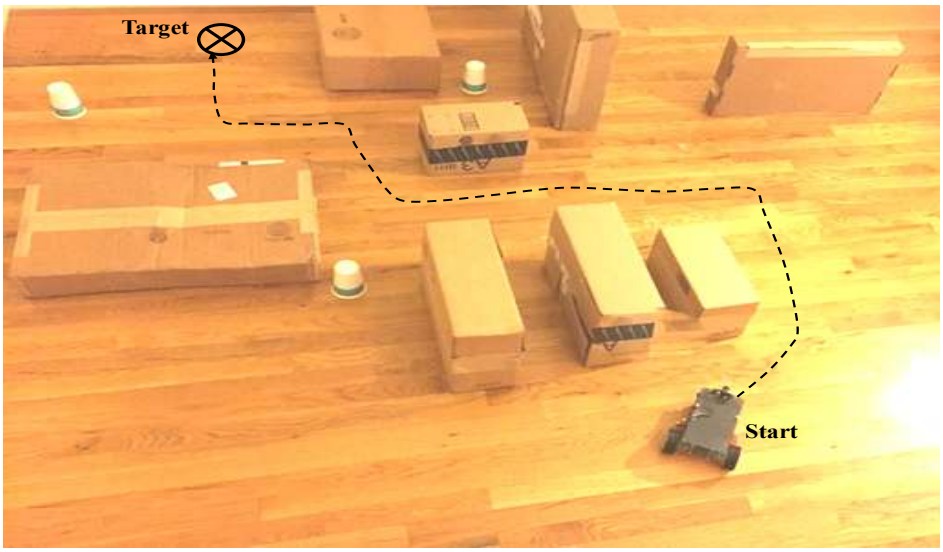

Figure.5.9. A snapshot of second real-time experiment for optimal path planning

The eight experiments were briefly summarized in table 5.2 to demonstrate the adaptability and reliability of our proposed model.

Table 5.2: The overview of eight experiments running to demonstrate the functional capabilities of the proposed model

Experiment	Sensory Devices	Experimental Platform	Results
1	Infrared Sensors	In high light frequency area with cylinders objects are placed around the robot examine the ability of infrared sensors to detects opaque obstacles.	The robot is able to autonomously navigate through obstacles and avoid any collision using two infrared sensors
2	Infrared sensors and Ultrasonic sensor	In high light frequency area with multiple translucent and transparent objects are placed around the robot examine how infrared sensors and ultrasonic sensor behave for different light frequencies that pass through objects.	As expected, the robot is able to detect all translucent and transparent objects are measure the distance between the robot and object.
3	Ultrasonic sensor	In dark area, to examine the ability of the ultrasonic sensor to detect obstacles.	The robot is able to autonomously navigate in dark area using ultrasonic sensor where the ambient light does not affect the results
4	Infrared Proximity Sensors	In dynamic environment with two robots, where each robot is equipped with proximity sensor to detect the other robot	The robot is able to detect moving object (another robot) to avoid any imminent collisions and measure their relative distance.
5	Camera	In dynamic environment, where two videos captured by camera are analyzed with each having different number of frames and objects.	The robot is able to detect obstacles as the detection rate for the first and second frames are 84.21% and 79.31% respectively .
6	Infrared reflective sensors, Infrared distance measuring sensors, Ultrasonic sensor, and Camera	In dynamic environment to detect U shaped obstacle.	Unlike other algorithms, the robot is able to detect U shaped obstacles and avoid collisions
7 and 8	Infrared reflective sensors, Infrared distance measuring sensors,	In dynamic environment, where multiple boxes and cups serves as obstacles..	The robot is able to smoothly navigate through different types of obstacles

	Ultrasonic sensor, and Camera	The robot started from given initial position and required to travel toward goal position using the shortest path.	and follow the optimal path (shortest path) from giving configuration to goal configuration

CHAPTER 6: RESULTS AND DISCUSSION

The main goal behind this study is to produce a low cost, low power consumption path planning and collision avoidance model for a wheeled mobile robot. The results of the seven experiments described earlier are discussed in the following subsections.

6.1. Infrared Distance Measuring Sensors

Infrared sensors used in this experiment has a detection range of 4 to 30 centimeters. Both sensors provide incorrect values for the distances out of their detection range (less than 4 cm and greater than 30 cm). The right infrared sensor reading is shown in Figure 6.1, which shows the infrared reading while the robot is navigating in the test field.

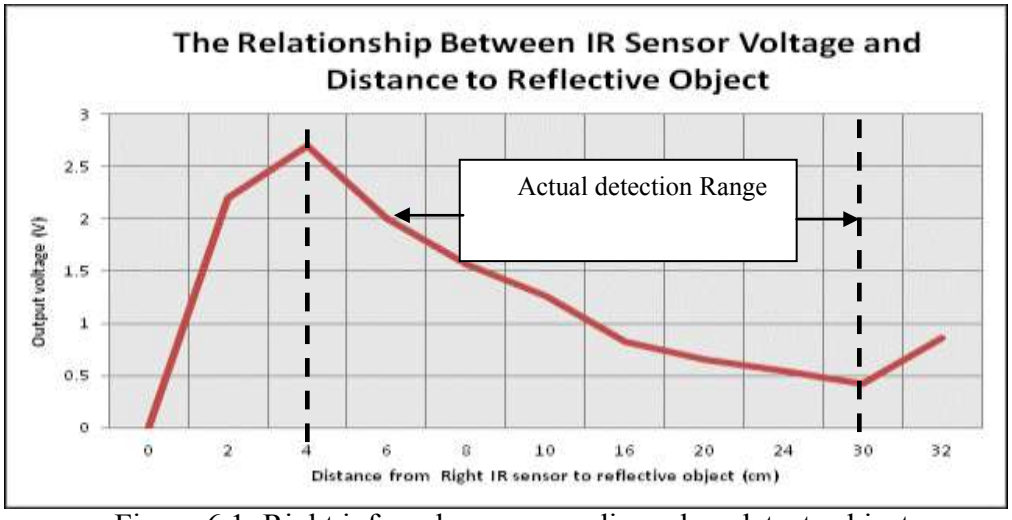

Figure 6.1: Right infrared sensors reading when detects object.

Figure.6.1, shows the relationship between the right infrared values and the distance to the reflective object. The output voltage of this sensor is dependent on the distance measured, as the distance increases, the output voltage decreases (inversely proportional) . This sensor offers an analogue voltage corresponding to the distance measured. Therefore, an analogue to digital converter (ADC) is required in order to find the distance from analog signals.

The ADC of our robot is a 10-bit ADC meaning it has the ability to detect 2^{10} analog levels, and the values returned by ADC are ratio-metric values, (ratio between the output voltage and analog levels). Thus:

$$ADC_{val} = \frac{ADC_{level} * Out_{vol}}{Opt_{vol}} \quad (6.1)$$

Where

- ADC_{level} :analog detecting level (= 1023 in our robot)
- Out_{vol} : analog output voltage at any given distance
- opt_{vol} : system operating voltage (= 5V in our robot)

In order to find the distance from analog values, the corrected inverse values of analog voltage related to ADC is computed in Figure 6.2.

As shown in Figure 6.2, the corrective constant value is added to the inverse of the distance, in order to nearly linearize the relationship as the corrective constant is the analog voltage output at distance 30 (equal to 0.42 in this experiment).

Then, a generalization can be figured and the fitting equation for the relationship between the distance and the voltage is as follows:

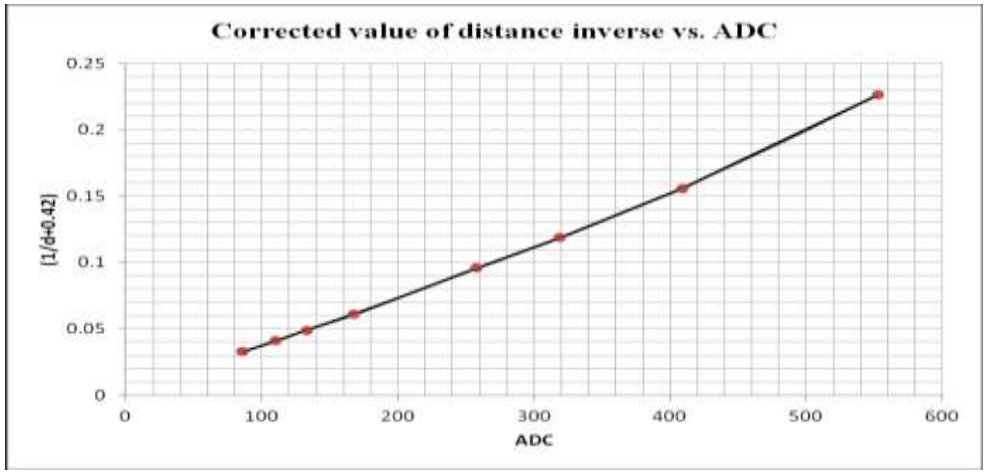

Figure 6.2**:** The relationship between corrected distance inverse with ADC.

$$\frac{1}{D+0.42} == a * ADC + b \quad (6.2)$$

where

- a is a constant parameter (=0.0004)
- b is a constant parameter (=- 0.0063)

In the case of ADC = 258.10

D = 1/ (0.0004 * 257.796 -0.0063) -0.42

D = 9.908 ≅10 As shown in Figure 6.3

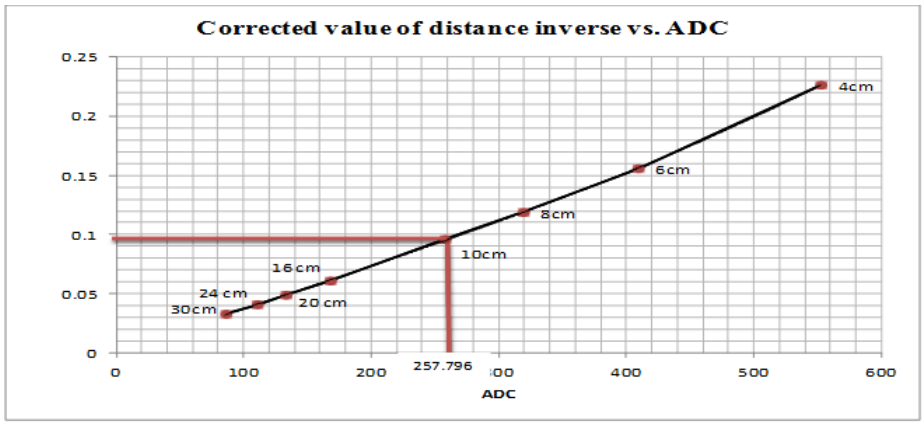

Figure 6.3: The corrected value of distance inverse vs. ADC

Table 6.1 shows the sensors readings at different times during the experiment.

Table 6.1: Infrared distance measuring sensors at different experimental times.

Time	RIR	LIR	
t_0	38.874	40.92	no object is detected
t_5	151.404	157.542	object is detected through RIR and LIR
t_{10}	20.46	28.644	no object is detected
t_{18}	40.92	42.966	no object is detected
t_{22}	216.876	40.92	object is detected through RIR
t_{31}	24.552	32.9406	no object is detected
t_{39}	36.828	43.3752	no object is detected
t_{41}	167.772	40.92	object is detected through RIR
t_{56}	26.598	20.6646	no object is detected
t_{64}	30.69	26.598	no object is detected
t_{72}	24.552	409.2	object is detected through LIR
t_{81}	36.828	30.69	no object is detected
t_{93}	40.92	257.796	object is detected through LIR
t_{102}	20.46	22.506	no object is detected
t_{118}	32.736	40.92	no object is detected
t_{126}	26.598	42.966	no object is detected

Sharp GP2D120 has a detection range between 4 to 30 cm and the corresponding ADC values ranges between 530 (for distance = 4 cm) and 80 (for distance = 30 cm). Both sensors return incorrect value for the distance out of their detection range(less than 4 cm and greater than 30 cm). For example, at time t_0, no object is detected as both sensors values are out of the detection range. At time t_5, both sensors values are greater than 80 and less than 530, which indicates that a front object is detected by

both sensors. At time $t_{22,}$ the right infrared sensor detects an object while the left infrared does not which indicates that the object is positioned at the front right of the robot.

6.2.Ultrasonic Sensor

The ultrasonic sensor is used to measures the distance to object in the range of 2 to 40 cm. The sensor was tested twice, and then compared with a real distance to ensure the validity and reliability of the sensor as shown in Figure.6.4.

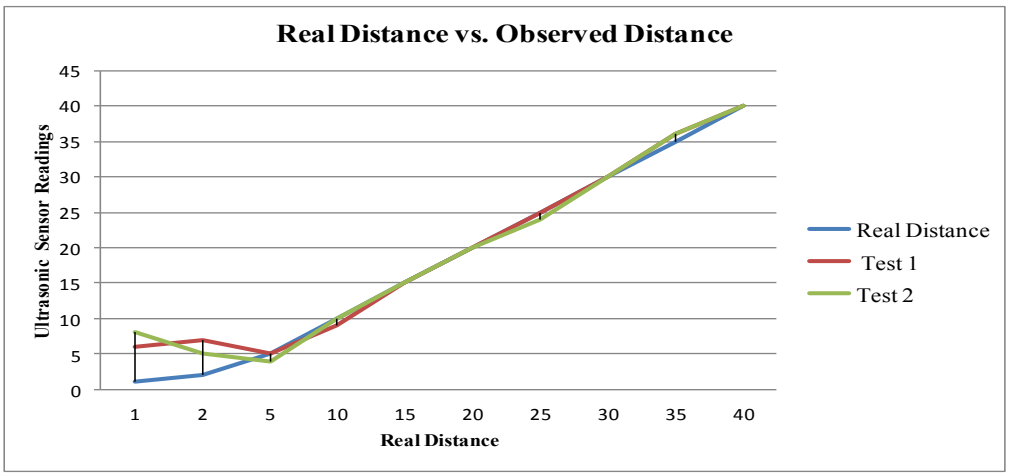

Figure 6.4: Ultrasonic sensor readings comparing with real distance

Figure.6.4, shows that by taking multiple readings for different distances, the ultrasonic sensor used is able to produce a fairly accurate representation of the object's location comparing with the real distance. However, values obtained by the ultrasonic sensor for distances out of its detection range(less than 2 centimeter) are not reliable.

6.3 Infrared and Ultrasonic sensors

Figure 6.5 shows the distance to object at different experimental time. The distance to object has been computed based on all sensors readings to determine the distance of the object from the robot.

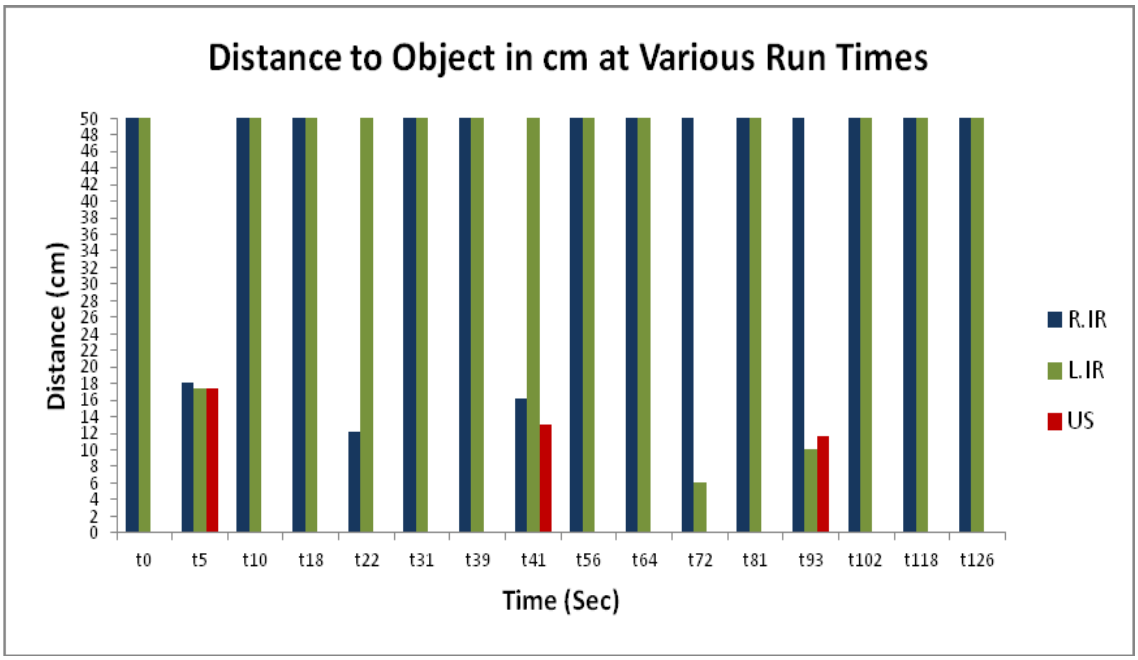

Figure 6.5 : Distance to object at different experimental time

As shown in Figure 6.5, at the initial position $t_{0,}$ no object is detected where the infrared distance measuring sensors produce invalid values and no distance is recorded for ultrasonic sensor. At time of t_5, a front object is detected through all sensors within the distance of 18 cm. At time of 41 seconds, a front right object is detected within the distance of 16 cm to right infrared sensor and 12 cm to ultrasonic sensor. At time of 72 seconds, left front object is detected through the left infrared sensor within the distance of 6.5 cm. At time $t_{93,}$ the robot detects another left front object as the left infrared

sensor returns distance = 10 cm and ultrasonic sensor return distance = 13 cm, while the right infrared sensor returns invalid measurement. The robot will continue moving forward while detecting objects and avoiding collisions along its path.

6.5. Obstacle Detection Scenarios

To characterize the performance of our method, multiple scenarios for obstacles are considered, where multiple obstacles are placed in different positions at the front of the robot as illustrated in Figure.6.6.

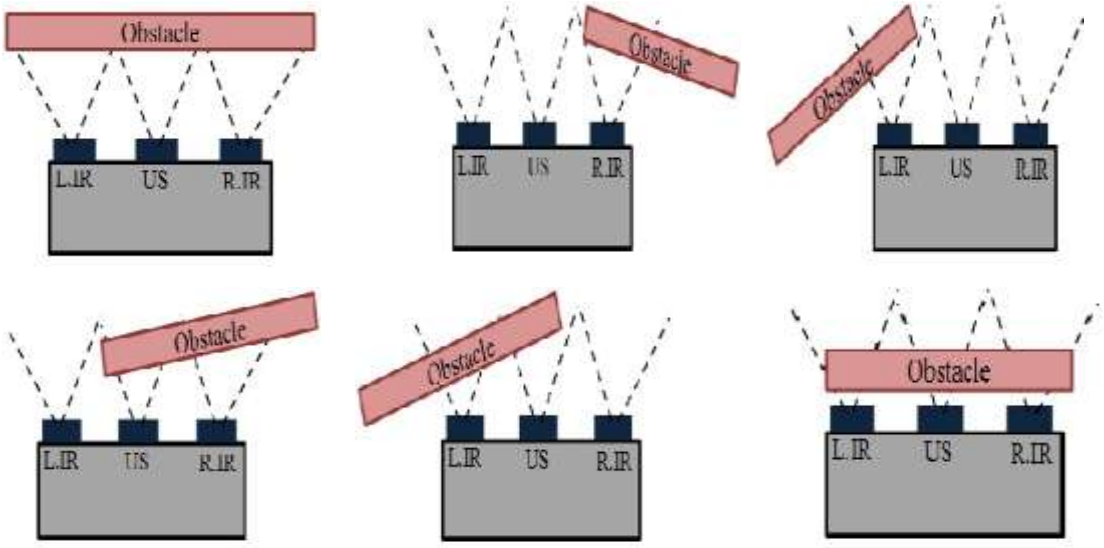

Figure 6.6: Obstacle detection scenarios

For different cases illustrated in Figure 6.6, sensors readings, distance to obstacles, and microcontroller decisions are provided in table 6.2.

Each two rows of table 6.2. represent one scenario, where it gives the detected sensor, sensors readings, distance to obstacles and the microcontroller decisions according to the sensors outputs. The infrared sensors produce an analog voltage output

between 3V (very close obstacle) and 0.4 V (far obstacle). Then, the ADC is needed to convert the analog voltage to measurable distance in centimeters using (1) and (2). Infrared sensors still return invalid values (less than 0.4 v), in the case of no obstacles, as they are affected by the light intensity of the tested environment. Moreover, the distance to object in cm can be obtain through the ultrasonic sensor (distance US3 module) when the function GetDistanceInCentimeters(US) is executed.

Table 6.2: Sensors readings and microcontroller decision at different experimental time

Time	**Sensor**	**R.IR**	**US**	**L.IR**	**Microcontroller Decision**
t0	None	0.2 V Invalid value	No value recorded	0.23 V Invalid value	Move forward
t1	R.IR US L.IR	0.74 V ADC: 151.404 D = 18 cm	D = 17.87 cm	0.74 V ADC: 151.40 D= 18 cm	Stop Turn
t2	None	0.12 V Invalid value	No value recorded	0.16 V Invalid value	Move forward
t3	R.IR	1.06 V ADC: 216.87 D= 12 cm	No value recorded	0.2 V Invalid value	Stop Turn left
t4	None	0.13 V Invalid value	No value recorded	0.10 V Invalid value	Move forward
t5	L.IR	0.12 V Invalid value	No value recorded	2 V ADC: 409.2 D= 5.93 cm	Stop Turn right
t6	None	0.20 V Invalid value	No value recorded	0.16 V Invalid value	Move forward
t7	US R.IR	0.82 V ADC: 167.772 D= 16 cm	D = 13.01 cm	0.24 V Invalid value	Stop Turn left
t8	None	0.13 V Invalid value	No value recorded	0.18 V Invalid value	Move forward
t9	L.IR US	0.20 V Invalid value	D = 11.72 cm	1.26 V ADC: 257.79 D= 9.90 cm	Stop Turn right
t10	None	0.19 V Invalid value	No value recorded	0.16 V Invalid value	Move forward
t11	IR Reflective	0.012V Invalid value	No value recorded	0.23 V Invalid value	Stop Turn
t12	None	0.22 V Invalid value	No value recorded	0.17 V Invalid value	Move forward

For example, At t_0, all sensors are activated, and the robot starts sensing the surroundings for obstacles. The two sharp sensors produce invalid values as there is no obstacle detected and no distance is recorded for the ultrasonic sensor. When t_1, a front obstacle is detected through all sensors, the output voltage from sharp sensors and Distance US3 module are converted to distances in cm. Then, according to the sensors readings; the robot stops for one second, backs up, turns and then move forward. At t_4, as an obstacle is detected through the right infrared sensor (R.IR) at distance 13 cm, the robot stops for one second, backs up turns right, and move forward. In the case of very close obstacle as in t_{12}, the robot detects the obstacle through the reflective sensors as they are used for edge detections and very small range obstacle detection. Then, the microcontroller makes the decision and avoid collision accordingly. The experimental findings demonstrates that our method provides a safer and smoother navigations in the presence of obstacles.

CHAPTER 7: ENERGY CONSUMPTION ANALYSIS

Mobile robots are limited by the finite amount of energy in the batteries they carry, since a new supply of energy while working is impossible, or at least too expensive to be realistic. Thus, the overall design of the multisensory system should mainly emphasize on enhancing the overall performance in terms of reducing power consumption. The total energetic utilization on mobile robot can be divided into two types: mobility energy and robotic energy. Mobility energy or " Motion energy" includes all of the power consumed, while the robot is moving to perform the assigned tasks. Thus, it is scaled with traveled distance, surface type and the robot velocity. On the other hand, the robotics energy includes all of the energy required to perform the tasks, so it is scaled with the total duty time and robots speed.

7.1Mobility energy

Motors are used to transform the electrical energy into mechanical energy to drive the robot. The mobility energy consumption is given by:

$$E_m = P_l + mas + mg\mu \qquad (7.1)$$

where P_l is transformation power loss by motor, m refers to robot mass, a is acceleration, s is robot's speed, g is gravity of earth, and μ is the ground friction. When the robot travels at high speed, the transforming loss and acceleration can be neglected

as they consume negligible energy compared with energy consumed by other modules. Consequently, the motion power is a linear of the speed. Several experiments were conducted to measure the impact of the robot's speed and distance traveled on the energy consumption. All the experiments were performed on a hardwood surface where a long cable was used to connect our robot with computer to measure the total energy consumption while the robot is moving on a straight line.

First experiment was conducted to examine the effects of the robot's speed on the amount of the energy consumed as shown in Figure 7.1. In this experiment, all other modules are turned off as the main purpose is to measure the amount of power consumed to keep the robot in motion. Moreover, this experiment was tested several times at different speed levels and each runs for 5 seconds.

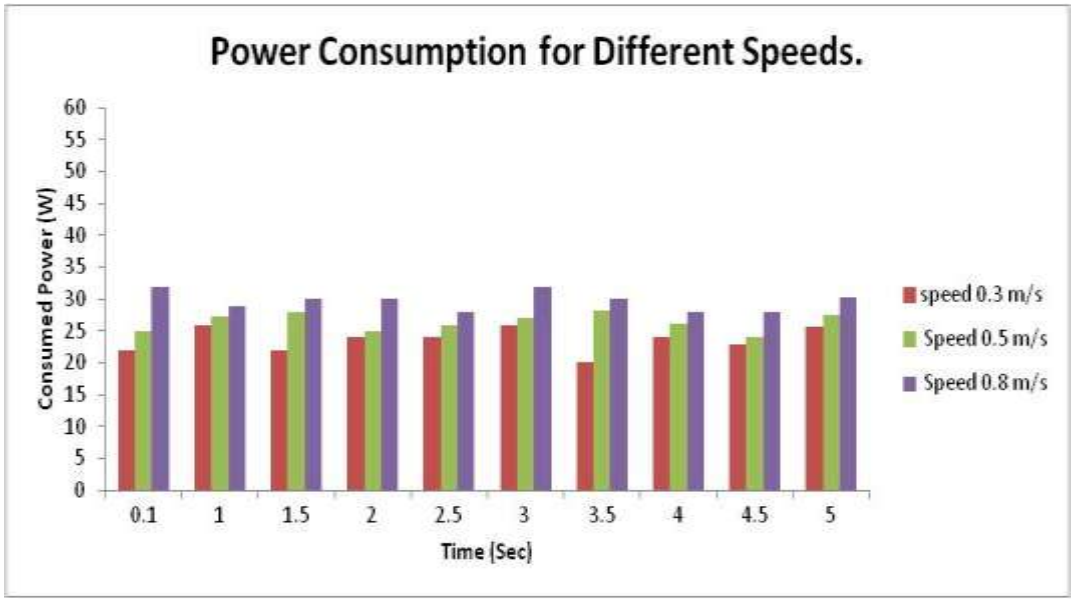

Figure 7.1: The power consumption at different speed level

As Figure 7.1, shows, the robot consumes more energy during the initializing steps to reach the highest specified speed, and then consumes less power as it moves on a straight path. By applying the least square method, a generalization can be drawn on the relationship between the robot's speed and the power consumption, as the speed increases, the power consumption increases.

The second experiment was carried to measure the impact of the travelling distance on power consumption. This experiment was tested two times for different distances (2 meters and 5 meters) as illustrated in Figure 7.2 .

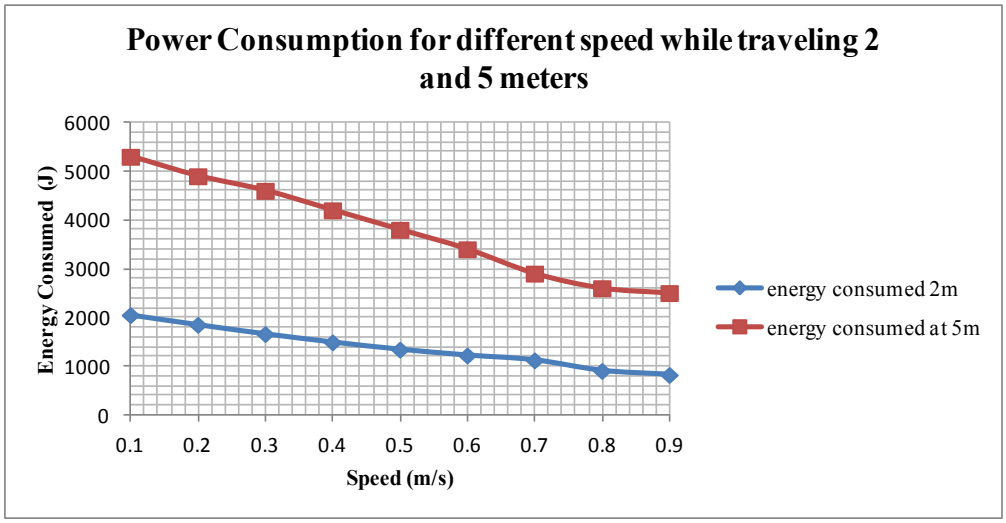

Figure 7.2:The total power consumption for different distances

Figure 7.2 shows that as the distance increases, the energy consumed increases as well. Furthermore, with a certain travelling distance, lower speed consumes more energy than at higher speed as more time is needed.

7. 2 Robotic energy

Robotic energy refers to all energy consumed to achieve assigned operation such as sensing, control, and computing energy. It consumes a significant amount of the total energy as the robot has to keep functioning whether it is moving or not. In the case of using different types of modules, it is important to measure the energy consumption for each component in order to promote the energy efficiency. The robotic energy is affected by robot's speed and the duty cycle. The operating duty cycle refer to the proportion of functioning time to the total operation time. Figure 7.3 shows the total robotics energy consumed to travel 2 meters at different duty cycles and different speed levels.

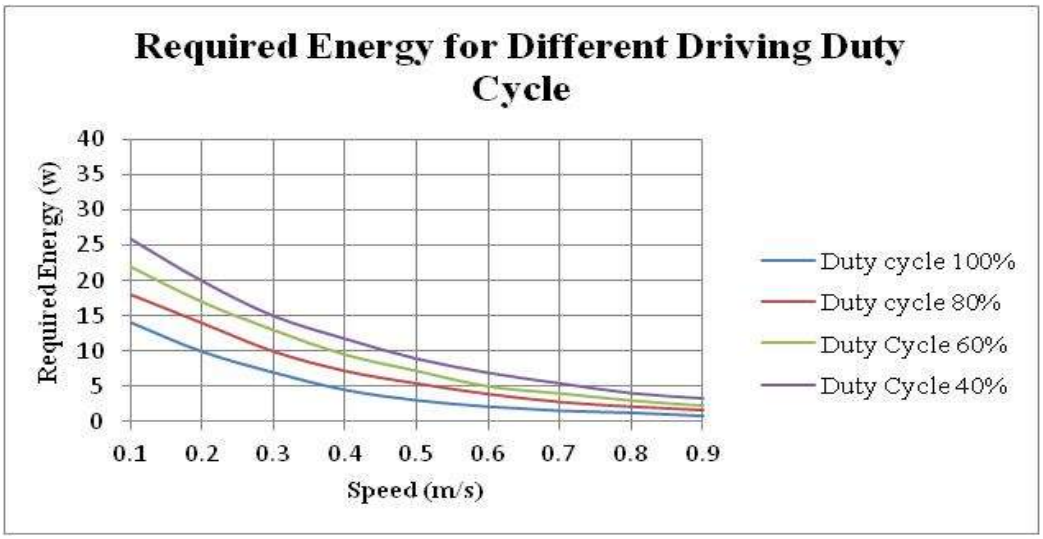

Figure 7.3:The power consumption at different duty cycles

Figure 7.3, shows that the energy level decreases as the duty cycle increases, in other words, largest duty cycles consumes less energy compared to other lower cycles.

Moreover, at each particular duty cycle, the power consumption increases as the speed level decreases.

The power consumption by the infrared and ultrasonic sensors are measured to improve the efficiency level and conserve some energy. The energy consumption by sensors is affected by the frequency level, the higher the frequency of a wave, the greater energy consumed. The power consumption of the two Infrared and ultrasonic sensors at different frequency levels is illustrated on Figure 7.4 and Figure 7.5 respectively.

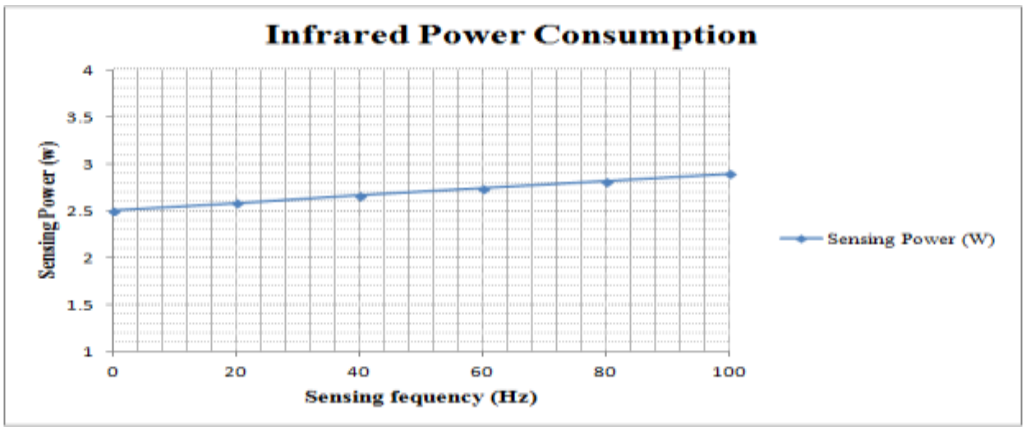

Figure 7.4: The power consumption by infrared sensor at different frequency levels

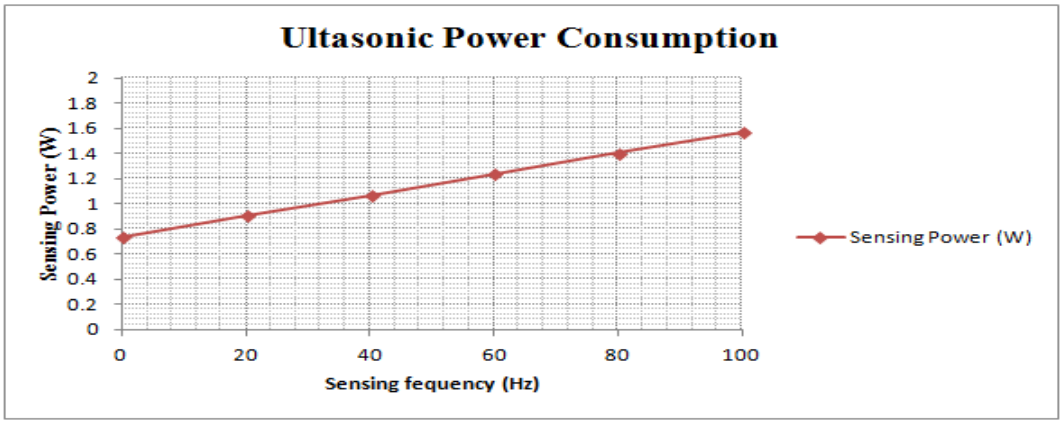

Figure 7.5: The power consumption by ultrasonic sensor at different frequency levels

Figure 7.6, summarizes the total power consumption on the robot including all the mobility energy and the robotic energy at maximum and minimum speed levels.

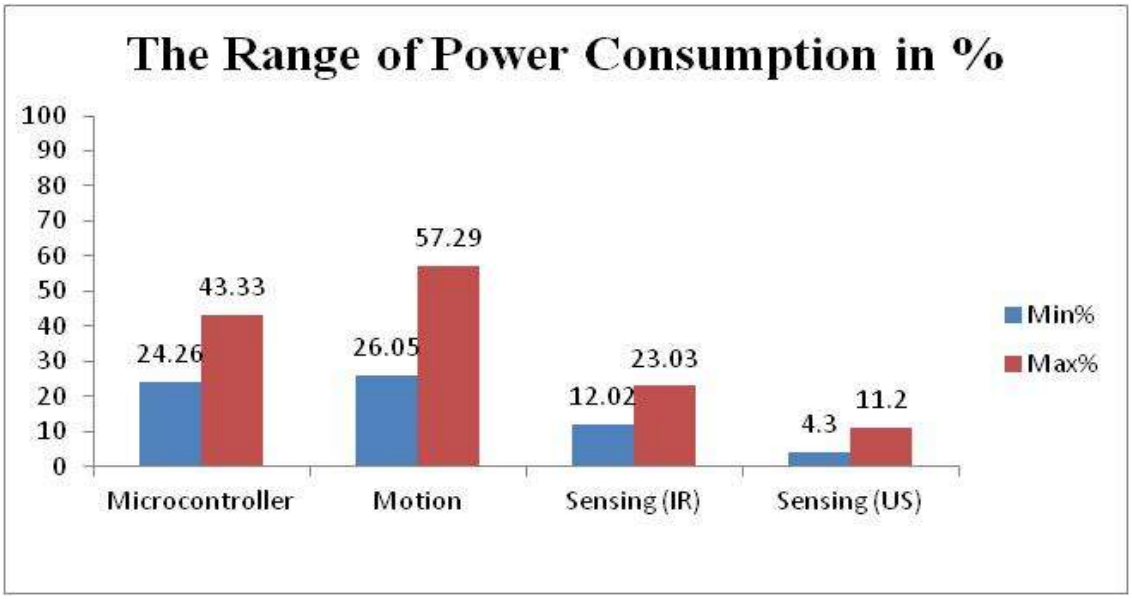

Figure 7.6: The range of power consumption by each component at maximum and minimum speed

CHAPTER 8: PERFORMANCE EVALUATION

A reliable path planning algorithm should feature a low computational complexity, less travelling distance, and smoother trajectory while avoiding collisions. Moreover, it must guarantee its safety in the dynamic environment while traversing toward the goal with as little cost and time as possible.

8.1 Computational analysis of algorithm

The collision-free path model used in this work is checking for all reachable solutions to destination and then follow the optimal one (shortest path) from the source point to destination point. The algorithm is exploring the state space by generating pointers of already-explored states in order to avoid expanding paths that are already expensive. The searching algorithm used can be implemented with a time complexity of ($n \log(n)$). Moreover, the algorithm uses a modified version of A^* to find the optimal reachable path to destination point [94]. The time to run the algorithm will be $(n \log(n) + n)$, thus proving the algorithm to be time and memory efficient compared to other algorithms discussed earlier in chapter 3.

Data acquisition (DAQ) was used to monitor the energy consumption while the robot is travelling from a given initial position to its goal and avoiding different types

of obstacles. To illustrate the results, further experiments were conducted with different number of detected obstacles as shown in Figure.8.1.

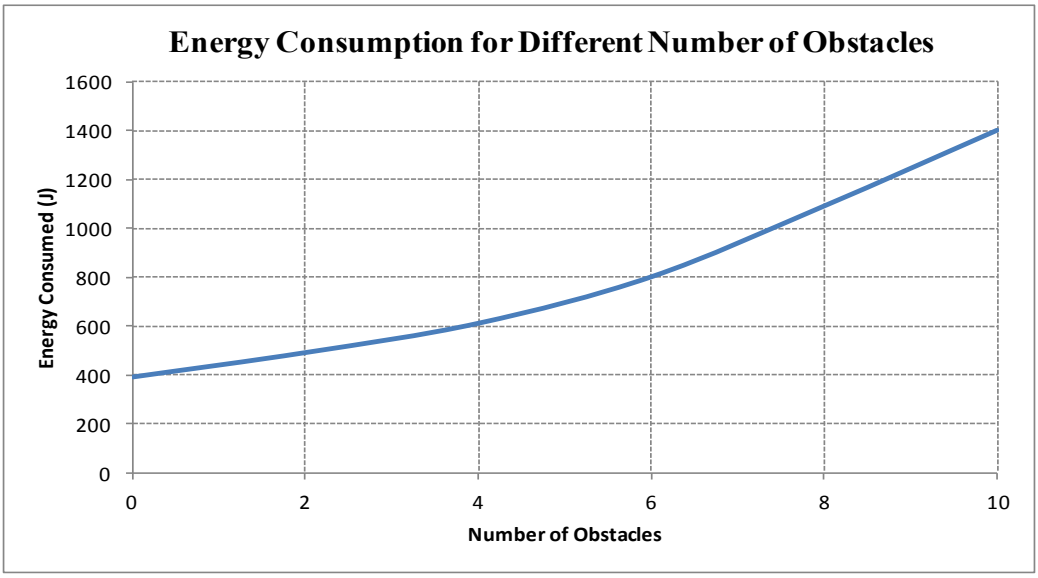

Figure.8.1 Energy Consumption for Different Number of Obstacles.

The energy consumption increases as the number of obstacles increased. The mobility energy is 397 (J) to cover 0.71 (m), which is the minimum distance from source to destination when no obstacles lies in the path [94].

The time-distance graphs of the proposed algorithm is depicted in Figure 8.2. and Figure 8.3 respectively. The minimum distance from a given initial position to the goal is 0.71 m, which is covered in 149 second when there are no obstacles detected along the path. Considering obstacles in the path, the time and travel distance will increase as the robot consumes more time and distance to spin around obstacles to avoid collisions [94].

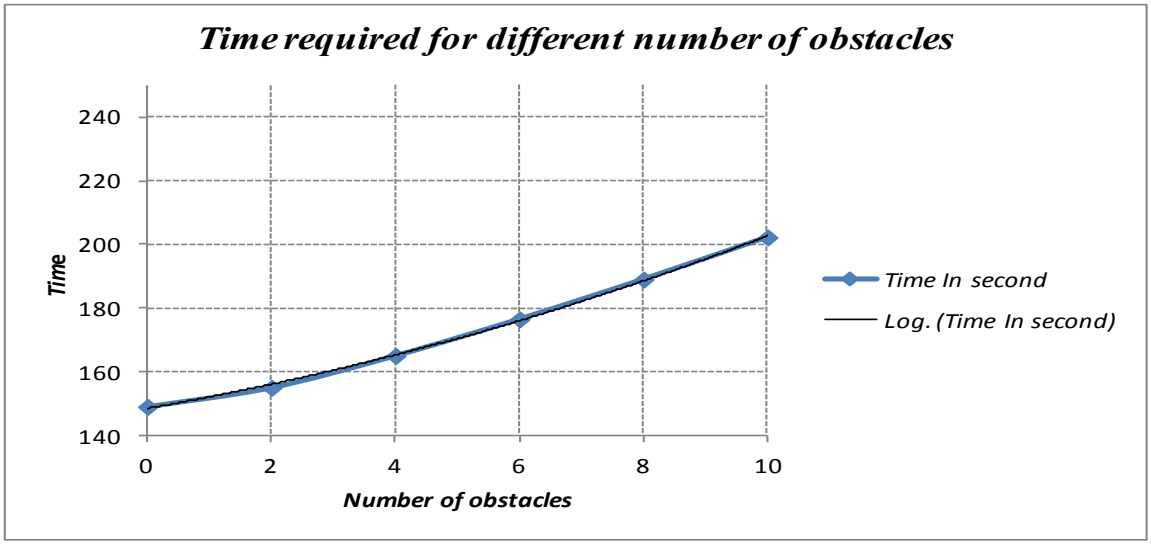

Figure.8.2. Path cost as a function of time for performance comparison

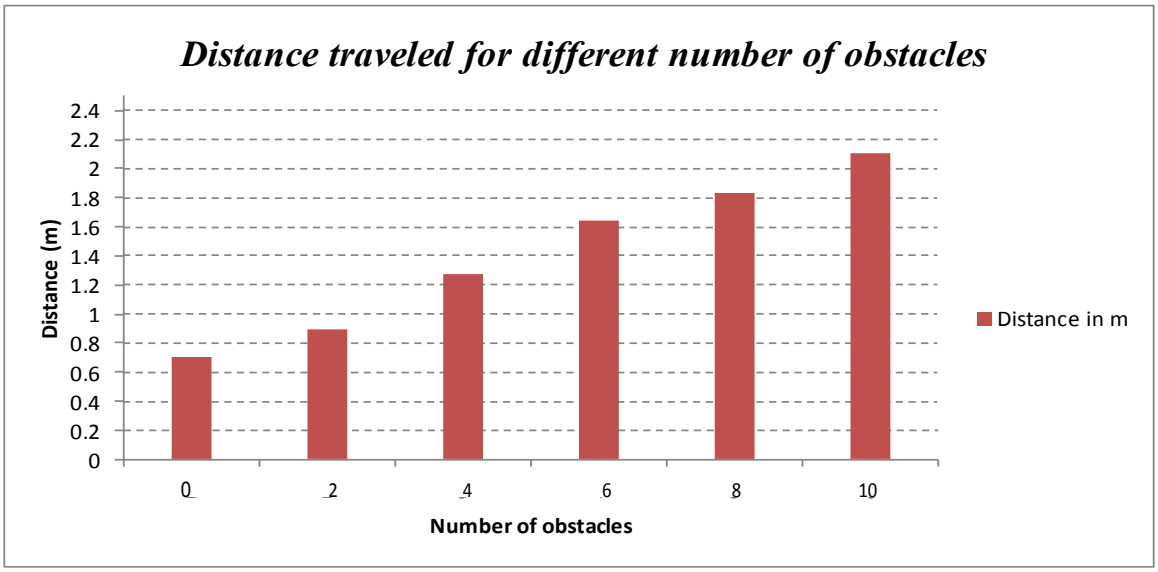

Figure.8.3. Path cost as a function of distance for different number of obstacles

8.2. Comparison and Evaluation

Obstacle avoidance in mobile robots depends on the robot's location, and sensors readings. There are several obstacle avoidance methods from simple navigating to

dynamic control strategy. The proposed method vary from others on the movement control and the use of sensorial data in order to avoid unexpected obstacles.

Table 8.1, summarizes how each of the obstacle avoidance algorithms identified earlier differ from each other by preferring the required hardware and overall performance. This section highlighted some significant findings, and focuses on the typical characteristics of each method. In the case of symmetric obstacles, FGM is more effective than APF. The symmetric obstacles are dead end scenarios in the APF, since the total force becomes zero and the robot stops moving, however, APF and FGM both fail for U shaped obstacles. VFH overcomes some of the APF restrictions; however, it requires more resources and calculations to generate a 2-dimensional grid and the conversion to 1-dimensional polar histogram. BA is easy to tune, but the efficiency level is too low as the robot may move too far from the destination. The proposed method uses low cost resources that does not require an external memory or high processor as in VFH. Moreover, it is not trapped into local minimum error as in APF, it is able to detect multiple obstacles in different shapes and colors. Furthermore, the proposed method can be used in real-time application with no need for a prior information about the surroundings, as the decision is based on the current reading of the sensors.

Table 8.1: Comparison between the proposed method and state-of-art methods

Method	Required modules	Time Complexity	Effectiveness	Comments
FGM [69]	Ultrasonic sensor, laser Sensors, Camera Processor	Obstacle avoiding using "FGM" is completed in three main steps which may increase the calculation time. *O(n (log n) ^{2})*	Always select shortest path, Able to avoid symmetric obstacles.	Difficult for Microcontroller as high computations are required. Fails for U shaped obstacles.
APF [90][92]	Ultrasonic sensor, IR distance Sensors, Microcontroller	Less time required as it selects shorter path. *O (n log n)*	Difficult to use in real-time application.	Symmetric obstacles is the dead end scenario for APF. Performs poorly on the correspondence of narrow passages
VFH [79] [93]	Ultrasonic sensor, High memory, Processor	It is computationally expensive (high memory and high processor are required). Required more time to generate a 2D grid and the conversion to 1D polar histogram	More resources required for calculation.	Since the world model is updated continuously, it is hard for microcontroller to perform these calculation. Select Shorter path comparing with other methods.
BA [75] [91]	Ultrasonic sensor, IR distance Sensors, Microcontroller	It has a unidirectional obstacle detection technique, which may increases the traversal time.	May take robot away from goal.	The robot may consume more time to reach goal as selects the longest path . Not goal-oriented, decision is based on the current sensor reading. No obstacles considered during edge detection process.
Proposed Method	IR sensors, Ultrasonic sensors Microcontroller	Low cost sensors Low computational power *O (n log n)*	Deployed in real-time. Detects obstacles in different shapes and colors.	Easy to tune And able to detect Symmetric and U shaped obstacles.

CHAPTER 9: CONCLUSION AND FUTURE WORK

In this Work, an integration of low cost infrared and ultrasonic sensors is presented to produce a complementary model of collision-free path for mobile robot. Multiple sensors are utilized together , so the benefits of one compensate for the limitations of the other in order to produce a reliable collision avoidance system.

The path planner with our model is utilized to generate the shortest path from a given initial position to a destination. Further experiments are performed to validate the effectiveness of the proposed model. To characterize the proposed model, multiple scenarios of obstacle-rich environments are considered, where infrared sensors readings and the microcontroller decisions are provided. The proposed model does not require prior information of the environment as the decision is based on the current percepts captured by the sensors. In addition, there is no need for high memory and a sophisticated processor, as a single microcontroller is enough to perform all computations. Furthermore, the proposed model, unlike other algorithms, can easily detects critical shaped obstacles (like U shape and H shape) which are considered dead-end scenarios for APF, VFH, and FGM algorithms.

The proposed model implemented and tested in a real mobile robot called FEZ-Cerbot from GHI electronics to validate the usefulness of the model. Moreover, time

complexity analysis and a comparison between the proposed method and state-of -art methods is presented to highlight some important findings, as it focuses on the typical characteristics of each method. The this model is characterized by its low cost, low power consumption and its usefulness in avoiding obstacles.

The field of collision-free path planning algorithms for a mobile robot in dynamic environments is one of the most well-studied problems that has a large scope. The model presented in this work can be extended in multiple promising directions. Future research has to be considered on the following points:

- The future work will comprise multiple scenarios to evaluate the overall performance of the proposed method. It will also focus on adding some other modules to enhance the efficiency and reliability of the model.
- In order to extend our study, the proposed method will be applied to resolve several problems such as multiple robots in dynamic environment for energy minimization to verify the validity and practicability of the method in unknown environments.
- Further improvement on the performance of the motion planning as it is limited by the basic model of computing the shortest path from a given point to goal point. Optimal algorithms for motion planning are to be derived to improve the motion planning performance.
- Several methods have been reported lately, to have combined the collision-free path planning algorithms with neural network and other machine learning

approaches. Considering a combination of these approaches with the proposed model is an important topic worthy of investigation.

REFERENCES

[1] R. Abiyev, D. Ibrahim, and B. Erin, "Navigation of mobile robots in the presence of obstacles," *Advances in Engineering Software*, vol. 41, no. 10-11, pp. 1179–1186, Oct. 2010.

[2] I. Ullah, Q. Ullah, F. Ullah, and S. Shin, "Integrated collision avoidance and tracking system for mobile robot," *International Conference of Robotics and Artificial Intelligence*, Oct. 2012.

[3] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers, "A hybrid collision avoidance method for mobile robots," *In Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1238–1243, May 1998.

[4] M. Bibuli, G. Bruzzone, M. Caccia, L. Lapierre, and E. Zereik, "A collision avoidance algorithm based on the virtual target approach for cooperative unmanned surface vehicles," *In Proceedings of the IEEE 22nd Mediterranean Conference on Control and Automation*, pp. 746–751, Jun. 2014.

[5] M. Tubaishat and S. Madria, "*Sensor networks: An overview*," IEEE Potentials, vol. 22, no. 2, pp. 20–23, Apr. 2003.

[6] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[7] R. Newton and M. Welsh, "Region streams: functional macroprogramming for sensor networks," *In the Proceedings of the 1st International Workshop on Data Management for Sensor Networks in Conjunction with VLDB 2004 - DMSN '04*, pp. 78–87, 2004.

[8] N. Marriwala and P. Rathee, "An approach to increase the wireless sensor network lifetime," *IEEE World Congress on Information and Communication Technologies*, pp. 495–499, Oct. 2012.

[9] S. Choochaisri, N. Pornprasitsakul, and C. Intanagonwiwat, "Logic Macroprogramming for wireless sensor networks," *International Journal of Distributed Sensor Networks*, pp. 1–12, 2012.

[10] U. Bischoff and G. Kortuem, "A state-based programming model and system for wireless sensor networks," *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07),* pp. 19–23, Mar. 2007.

[11] A. Alajlan and K. Elleithy, "High-level abstractions in wireless sensor networks: Status, taxonomy, challenges, and future directions," *Proceedings of the Zone 1 Conference of the American Society for Engineering Education*, Apr. 2014.

[12] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, May 2005.

[13] R. Venkateswarlu and D. Janakiram, "A simple model for evaluating the Scalability in wireless sensor networks," *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Dec. 2005.

[14] M. Sousa, A. Kumar, M. Alencar , W. Lopes, “Scalability in An Adaptive Cooperative System for Wireless Sensor Networks,” *IEEE International Conference on Ultra-Modern Telecommunications Workshops (ICUMT)* , Oct. 2009.

[15] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[16] Z. Chaczko, R. Klempous, J. Nikodem, M. Nikodem, J. Rozenblit, "An Improvement of Energy Aware Routing in Wireless Sensors Network", *European Modeling and Simulation Symposium, Barcelona*, Oct. 2006.

[17] Y. Takizawa, "Node localization for sensor networks using self-organizing maps," *IEEE Topical Conference on Wireless Sensors and Sensor Networks*, pp. 61–64, Jan. 2011.

[18] S. Pandey, P. Prasad, P. Sinha, and P. Agrawal, "Localization of sensor networks considering energy accuracy tradeoffs," *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1–10, Dec. 2005.

[19] M. Rudafshani and S. Datta, "Localization in wireless sensor networks," *IEEE 6th International Symposium on Information Processing in Sensor Networks*, Apr. 2007.

[20] M. Bellalouna and A. Ghabri, "A priori methods for fault tolerance in wireless sensor networks," *World Congress on Computer and Information Technology (WCCIT)*, Jun. 2013.

[21] L. Wang, Jian-feng, C. Wang, and A. C. Kot, "Fault and intrusion tolerance of wireless sensor networks," *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, 2006.

[22] R. Sugihara and R. K. Gupta, "Programming models for sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 2, pp. 1–29, Mar. 2008.

[23] D. D. Geetha, N. Nalini, and R. C. Biradar, "Active node based fault tolerance in wireless sensor network," *Annual IEEE India Conference (INDICON),* Dec. 2012.

[24] A. Alajlan, B. Dasari, Z. Nossire, K. Elleithy, and V. Pande, "Topology management in wireless sensor networks: Multi-state Algorithms," *International Journal of Wireless & Mobile Networks*, vol. 4, no. 6, pp. 17–26, Dec. 2012.

[25] D. D. Chaudhary and L. M. Waghmare, "Energy efficiency and latency improving protocol for wireless sensor networks," *International Conference on Advances in Computing, Communications and Informatics (ICACCI),* pp. 1303–1308, Aug. 2013.

[26] K. Vardhe, C. Zhou, and D. Reynolds, "Energy efficiency analysis of multistage cooperation in sensor networks," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1–5, Dec. 2010.

[27] M. Huang and Y. H. Hu, "Collaborative sampling in wireless sensor networks," IEEE *Global Telecommunications Conference (GLOBECOM)*, pp. 1–5, Dec. 2010.

[28] W. Li, J. Bao, and W. Shen, "Collaborative wireless sensor networks: A survey," *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2614–2619, Oct. 2011.

[29] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61–72, Mar. 2002.

[30] A. Alajlan and K. Elleithy, "Programming Models for Wireless Sensor Networks: Status, Taxonomy, Challenges, and Future Directions," *International Journal of Scientific and Engineering Research(IJESR)*, vol. 6, no. 5, May 2015.

[31] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 5, pp. 93–104, Dec. 2000.

[32] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," *ACM SIGPLAN Notices*, vol. 38, no. 5, pp. 1-11, May 2003.

[33] K. Römer, O. Kasten, and F. Mattern, "Middleware challenges for wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 59–61, Oct. 2002.

[34] M.-M. Wang, J.-N. Cao, J. Li, and S. K. Dasi, "Middleware for wireless sensor networks: A survey," *Journal of Computer Science and Technology*, vol. 23, no. 3, pp. 305–326, May 2008.

[35] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for Robotics: A Survey," *In Proceedings of The IEEE International Conference on Robotics, Automation, and Mechatronics* (RAM), pp. 736–742, Sep. 2008.

[36] A. Makarenko, A. Brooks, and T. Kaupp, "Orca: Components for Robotics," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 163-168, Oct. 2006.

[37] B. Rubio, M. Diaz and J. Troya, “Programming Approaches and Challenges for Wireless Sensor Networks,” *IEEE Second International Conference on Systems and Networks Communications (ICSNC),* Aug. 2007.

[38] P. Levis and D. Culler, "Maté: A Tiny Virtual Machine for Sensor Networks," *In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X),* pp. 85–95, Oct. 2002.

[39] P. Levis, D. Gay, and D. Culler, “Active Sensor Networks,” *In Proceedings of the 2nh International Symposium on Networked Systems Design and Implementation (NSDI’05),* pp. 29–42, Mar. 2005.

[40] T. Liu and M. Martonosi, "Impala: a middleware system for managing autonomic, parallel sensor systems," *In Proceedings of the 9th ACM SIGPLAN symposium on Principles and practice of parallel programming ACM SIGPLAN Notices*, vol. 38, no. 107, p. 118, Jun. 2003.

[41] L. Mottola and G. P. Picco, "Programming wireless sensor networks with logical neighborhoods," *In Proceedings of the first international conference on Integrated internet ad hoc and sensor networks,* May. 2006.

[42] P. A. Vicaire, E. Hoque, Z. Xie, and J. A. Stankovic, "Bundle: A group-based programming abstraction for Cyber-Physical systems," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 379–392, May 2012.

[43] R. Gummadi, O. Gnawali, and R. Govindan, "Macro-programming wireless sensor networks using Kairos," in Distributed Computing in Sensor Systems. Springer Nature, pp. 126–140, July. 2005.

[44] E. Brewer, D. Culler, D. Gay, P. Levis, R. von Behren, and M. Welsh, "NesC: A programming language for deeply Networked systems," 2004. [Online]. Available: http://nescc.sourceforge.net/. Accessed: Dec. 6, 2015.

[45] "TinyOS home page,". [Online]. Available: http://www.tinyos.net/. Accessed: Dec. 6, 2015.

[46] P. Levis, D. Gay, and D. Culler, "Active Sensor Networks," *In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, vol. 2, pp. 343–356, May. 2005.

[47] L. Mottola and G. P. Picco, "Programming wireless sensor networks," *ACM Computing Surveys*, vol. 43, no. 3, pp. 1–51, Apr. 2011.

[48] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler, "Hood: a neighborhood abstraction for sensor networks," *In Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pp. 99–110, Jun. 2004.

[49] M.Welsh and G.Mainland, "Programming Sensor Networks Using Abstract Regions," In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, vol.1, Mar. 2004.

[50] T. Abdelzaher et al., "EnviroTrack: Towards an environmental computing paradigm for distributed sensor networks," In *Proceedings of the 24th International Conference on Distributed Computing Systems*, Mar. 2004.

[51] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," *In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 263–270, Aug. 1999.

[52] L. Mottola and G. P. Picco "Logical Neighborhoods: A Programming Abstraction For Wireless Sensor Networks", *In Proceedings of the 2nd International Conference on Distributed Computing on Sensor Systems (DCOSS)*, pp.150 -168, 2006.

[53] R. Newton, G. Morrisett, and M. Welsh, "The regiment Macroprogramming system," *In Proceedings of the* 6th *International Symposium on Information Processing in Sensor Networks*, Apr. 2007.

[54] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[55] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *ACM SIGMOD Record*, vol. 31, no. 3, pp. 9-18, Sep. 2002.

[56] W. F. Fung, D. Sun, and J. Gehrke, "COUGAR: The Network is the Database," *In Proceedings of the ACM SIGMOD international conference on Management of data*, Jun. 2002.

[57] D. Johnson et al., "Mobile Emulab: A robotic wireless and sensor network Testbed," *In Proceedings of the 25TH IEEE International Conference on Computer Communications*, pp. 1–12, Apr. 2006.

[58] R. Luna, A. Oyama, KE, Bekris, " Network-Guided Multi-Robot Path Planning for Resource-Constrained Planetary Rovers," *In Proceedings of the 10th International Symposium on Artificial Intelligence, Robotic and Automation Space*, Aug.2010.

[59] Z. Yao and K. Gupta, "Distributed Roadmaps for robot navigation in sensor networks," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 997–1004, Oct. 2011.

[60] J. Zheng and A. Jamalipour, Eds., "Chapter2: network architecture and protocol stack," in Wireless sensor networks: A Networking Perspective. Wiley-Blackwell, Jan. 2009.

[61] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, Introduction to Autonomous Mobile Robots, vol. 19, Second Edition ed. MIT Press, , pp.379–380, Dec. 2004.

[62] R. Saputra, E. Rijanto and H. Saputra,"Trajectory Scenario Control for the Remotely Operated Mobile Robot LIPI Platform Based on Energy Consumption Analysis," *International Journal of Applied Engineering Research*, vol. 7, no. 8, pp. 851-866, 2012.

[63] D. Fox, W. Burgard, S. Thrun, "A hybrid collision avoidance method for mobile robots", In *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1238-1243, 1998.

[64] R. Kurzweil," *The Ages of Intelligence Machines*" MIT Press, Cambridge, Massachusetts 1990.

[65] E. Rich and K. Knight. "*Artificial Intelligence*" McGraw-Hill, New York, NY, 2nd edition, 2009

[66] K.M. Passino, "Intelligent Control: An Overview of Techniques," *in Perspectives in control engineering: Technologies, applications, and new directions*, T. Samad, Ed. NJ: IEEE Press, 2001.

[67] A. Benitez, D. Vallejo, G. de los Santos, and E. Medrano, "New technique to build probabilistic roadmap methods," In *Proceedings of the 14th International Conference on Electronics, Communications and Computers*, 2004.

[68] H. Alt and E. Welzl, "Visibility graphs and obstacle-avoiding shortest paths," Mathematical Methods of Operations Research, vol. 32, pp. 145-164, 1988.

[69] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: 'Follow the gap Method'," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, Sep. 2012.

[70] J.Oroko, B.Ikua, "Obstacle Avoidance and Path Planning Schemes for Autonomous Navigation of a Mobile Robot: A Review". *Sustainable Research and Innovation Proceedings*, Vol. 4, 2012.

[71] D. Fu-guang, J. Peng, B. Xin-qian, and W. Hong-jian, "AUV local path planning based on virtual potential field," IEEE International Conference Mechatronics and Automation, Aug. 2005.

[72] A. Babinec, M. Dekan, F. Duchoň, and A. Vitko, "Modifications of VFH navigation methods for mobile robots," *Procedia Engineering*, vol. 48, pp. 10–14, 2012.

[73] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, Jun. 1991.

[74] Z. Yan, Y. Zhao, S. Hou, H. Zhang, and Y. Zheng, "Obstacle avoidance for unmanned undersea vehicle in unknown unstructured environment," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–12, 2013.

[75] J. Ng and T. Bräunl, "Performance comparison of bug navigation Algorithms," Journal of Intelligent and Robotic Systems, vol. 50, no. 1, pp. 73–84, April. 2007.

[76] A. Yufka and O. Parlaktuna, "Performance comparison of BUG Algorithms for mobile robots," *In Proceedings of the 5th International Advanced Technologies Symposium*, pp. 61–65, May. 2009.

[77] S. Michaud et al., "Solero: Solar-Powered Exploration Rover," *In Proceedings of the 7th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, 2002.

[78] S. E. Mushi, Z. Lin, and P. E. Allaire, "Design, construction, and modeling of a flexible rotor active magnetic bearing test rig," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 6, pp. 1170–1182, Dec. 2012.

[79] S. Liu and D. Sun, "Minimizing energy consumption of wheeled mobile robots via optimal motion planning," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp. 401–411, Apr. 2014.

[80] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic Algorithms for path-planning optimization in autonomous mobile robots," *Soft Computing*, vol. 11, no. 3, pp. 269–279, Mar. 2006.

[81] N. Ganganath, C.-T. Cheng, and C. K. Tse, "Finding energy-efficient paths on uneven terrains," *In Proceedings of the 10th France-Japan/ 8th Europe-Asia Congress on Mecatronics*, Nov. 2014.

[82] Y. Mei, Y.-H. Lu, C. S. G. Lee, and Y. C. Hu, "Energy-efficient mobile robot exploration," *In Proceedings of the IEEE International Conference on Robotics and Automation*, May 2006.

[83] H. Kim and B. K. Kim, "Minimum-energy trajectory generation for cornering with a fixed heading for Three-Wheeled Omni-Directional mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 75, no. 2, pp. 205–221, Jun. 2013.

[84] J. E. Mohd Salih, M. Rizon, and S. Yaacob, "Designing Omni-Directional mobile robot with Mecanum wheel," *American Journal of Applied Sciences*, vol. 3, no. 5, pp. 1831–1835, May 2006.

[85] J. Brateman, C. Xian, and Y. Lu, "Energy-Effcient scheduling for autonomous mobile robots," *In Proceedings of the IFIP International Conference on Very Large Scale Integration*, Oct. 2006.

[86] M. Chemnitz, G. Schreck, and J. Kruger, "Analyzing energy consumption of industrial robots," *In Proceedings of the IEEE 16th Conference on Emerging Technologies & Factory Automation*, Sep. 2011.

[87] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the Heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[88] A. Alajlan, M. Almasri, and K. Elleithy, "Multi-sensor based collision avoidance algorithm for mobile robot," *IEEE Long Island Systems, Applications and Technology (LISAT)*, May 2015.

[89] "GHI Electronics,". [Online]. Available: https://www.ghielectronics.com. Accessed: Jan. 7, 2015.

[90] A. Sgorbissa and R. Zaccaria, "Planning and obstacle avoidance in mobile robotics," *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 628–638, Apr. 2012.

[91] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," *In Proceedings of the Congress on Evolutionary Computation*, Jul. 2000.

[92] I. Susnea, A. Filipescu, G. Vasiliu, G. Coman, and A. Radaschin, "The bubble rebound obstacle avoidance algorithm for mobile robots," *In Proceedings of the 8th IEEE International Conference on Control and Automation*, Jun. 2010.

[93] S. R. Lindemann and S. M. LaValle, "Simple and efficient Algorithms for computing smooth, collision-free feedback laws over given cell Decompositions," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 600–621, May 2009.

[94] A. Alajlan and K. Elleithy, " Low Power Consumption, Low-Cost Multisensory Based System for Autonomous Navigational Mobile Robot," *IEEE 42nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* March 2017.

PUBLICATIONS

Journal Publications

- Abrar Alajlan, Benjamin Dasari, Zyad Nossire, Khaled Elleithy and Varun Pande Topology Management In Wireless Sensor Networks: Multi-State Algorithms, *International Journal of Wireless & Mobile Networks (IJWMN),* On page(s) 17-26 Vol 4 (6), Dec 2012.
- Abrar Alajlan, Khaled Elleithy, and Varun Pande, "A New Programming Model to Simulate Wireless Sensor Networks : Finding The Best Routing Path," International Journal of Compputer Networks & Communications (IJCNC, Vol. 5., No. 4, pp. 1 – 14, July 2013.
- Abrar Alajlan, Khaled Elleithy, "Programming Models for Wireless Sensor Networks: Status, Taxonomy, Challenges, and Future Directions "*International Journal of Scientific and Engineering Research(JESR) Vol.6 (5),*May. 2015.
- Alajlan, A., Elleithy, K., Almasri., M "A Low Cost, Low Power Consumption, Multi-Sensor Collision Avoidance Algorithm for Autonomous Mobile Robots ". Submitted to *Journal of Intelligent & Robotic Systems* , 2016 (Impact Factor 1.178)
- Almasri, M., Elleithy, K., & Alajlan, A. (2015). "Sensor Fusion Based Model for Collision Free Mobile Robot Navigation". *Sensors,* 16(1), 24. (Impact Factor 2.24)
- Marwah Almasri, Abrar M. Alajlan, and Khaled M. Elleithy, "Trajectory Planning and Collision Avoidance Algorithm for Mobile Robotics System", *IEEE Sensors Journal* (Impact Factor 1.762)

Conference Publications

- Abrar Alajlan, Khaled Elleithy, " Low Power Consumption, Low-Cost Multisensory Based System for Autonomous Navigational Mobile Robot", *IEEE 42nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2017),* March 5-9, 2017.
- Abrar Alajlan, Marwah M. Almasri, Khaled M. Elleithy, "Multi-Sensor Based Collision Avoidance Algorithm for Mobile Robot", *IEEE Long Island Systems, Applications and Technology Conference*, March 2015. (*Best Paper Award*).
- Abrar Alajlan, Khaled Elleithy ," High-Level Abstractions in Wireless Sensor Networks: Status, Taxonomy, Challenges, and Future Directions, *ASEE Northeast Section Conference*, Bridgeport. CT, April 4-5, 2014.
- Almasri,M. , Elleithy,K., Alajlan.A;"Development of Efficient Obstacle Avoidance and Line Following Mobile Robot with the Integration of Fuzzy Logic System in Static and Dynamic Environments", *IEEE Long Island Systems, Applications and Technology Conference*, April 2016.

Posters

- Alajlan, A., Elleithy, K., Almasri., M "Multi-Sensor Dynamic Motion Planning and Collision Avoidance for Autonomous Mobile Robot" *Annual IEEE Connecticut*

Conference on Industrial Electronics, Technology & Automation (CT-IETA 2016) at University of Bridgeport, Oct 14. (*Honorable Mention Award).*

- Marwah Almasri, Khaled Elleithy, and Abrar Alajlan, "Path Planner Mobile Robot with Collision Avoidance Technique," Poster presentation at *Annual IEEE Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA 2016),* University of Bridgeport, Oct 14, 2016. *(Honorable Mention Award).*
- Alajlan, A., Elleithy, K., Almasri., M "Energy-Efficient Dynamic Motion Control for Wheeled Mobile Robots Using Low Cost Resources" poster presented as part of *Northeast Section ASEE Conference* at University of Rhode Island, April 28 - 30, 2016. *(4^{th} Place Award) (Displayed at a reception in Bridgeport city hall hosted by Bridgeport Mayor April 7)*
- Almasri,M., Elleithy,K., Alajlan. A " Fuzzy Logic Control for Autonomous Mobile Robots in Static and Dynamic Environments" poster, *Faculty Research Day(FRD)* at University of Bridgeport, April *1^{st} 2016 (3^{rd} Place Award)*
- Almasri,M., Alajlan.A, Elleithy,K;" Collision Avoidance Algorithm for Multi-sensor Mobile Robot " poster presented as part of *Northeast Section ASEE Conference* at Northeastern University, April 30 – May 2, 2015.
- Abrar Alajlan, Khaled Elleithy;: High-Level Abstractions in Wireless Sensor Networks: Status, Taxonomy, Challenges, and Future Directions", *ASEE Northeast Section Conference,* Northfield, CT, April 4-5, 2014.